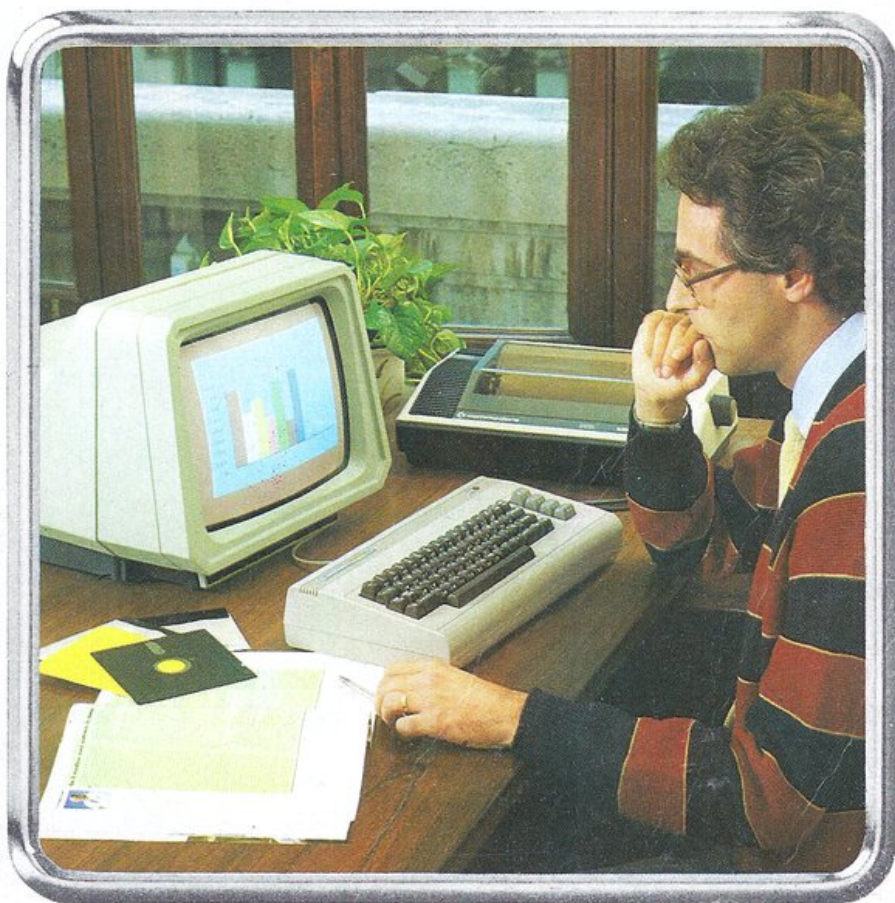
I MANUALI DI

20 PROGRAMMI PER

COMMODORE 64

- Poker d'assi
- Un medico poco serio
- Computo, computas...
- Strike and ball
- La torre di Hanoi
- Grafica monodimensionale
- Corsa d'auto
- Alta risoluzione
- Disegna il tuo sprite
- Grafica tridimensionale
- Slot machine
- Proviamo con l'Assembler
- Labirinto
- Il gioco della vita
- Memory dump
- Archivio
- Aritmetica a precisione infinita
- L'archivio veloce
- Editor
- Le otto regine

ARMANDO CURCIO EDITORE

I MANUALI DI BASIC

20 PROGRAMMI PER

COMMODORE 64

Collana realizzata dalla Divisione Opere Scientifiche
della Armando Curcio Editore

a cura di **Maurizio Talamo** e **Mario Tozzoli**

consulenza software
Franco Arcieri

Direzione Editoriale	**Gabriella Costarelli**
Redattore Capo	**Marcella Marcaccini**
Coordinamento redazionale	**Ugo Spezia**
Segreteria di redazione	**Concetto Albergo, Maurizio Pinto**
Revisione	**Massimo Rossi, Sandro Sandri**
Correzione	**Maria Albergo, Anna Landi, Gina Melone, Rita Tancredi, Graziella Tassi**
Ricerca iconografica	**Carla Bertini, Rossella Pozza**
Grafica e Direzione Artistica	**Vittorio Antinori**
Realizzazione grafica	**Roberto Sed, Riccardo Catani**
Disegni e grafici	**Renato Lazzarini, Gianni Mazzoleni Rolando Mazzoni, Francesco Izzo**
Produzione	**Piergiorgio Palma**
Segreteria di produzione	**Rita Costa**
Archivio	**Stefano Bechelloni, Ulla Näslundh**

Fotocomposizione Tipocrom, Roma. Stampa: O.F.S.A., Casarile, Milano

20 PROGRAMMI PER COMMODORE 64

ARMANDO CURCIO EDITORE

Introduzione

Questo volumetto, ancor prima di costituire una pura e semplice raccolta di interessanti programmi di utilità e di gioco, è una guida didattica alla programmazione in linguaggio Basic del personal computer Commodore 64.
Chi si avvicina per la prima volta ad un computer non deve essere costretto ad esercitare sulla tastiera la propria immaginazione, oltre che le proprie dita, per capire come funziona e a cosa serve ciò che ha di fronte: una preventiva illustrazione delle funzionalità associate ai tasti può risparmiare ore di tentativi, il più delle volte sterili e destinati a non lasciare un'idea chiara di ciò che si è fatto in realtà.
Per questi motivi si è ritenuto utile, dopo una descrizione della configurazione del sistema Commodore 64, passare ad un accurato e dettagliato esame della tastiera, impiegando estesamente la parte illustrativa per facilitare l'individuazione dei singoli tasti e per insegnare ad attivarli nelle diverse modalità di funzionamento.
Anche l'impiego del video può presentare difficoltà che sono spesso sottovalutate da chi redige i manuali ufficiali che corredano l'hardware. Nel paragrafo dedicato al video sono illustrati con particolare attenzione le funzioni di controllo del cursore e l'impiego delle notevoli possibilità cromatiche del Commodore 64, utilizzando ancora una volta estesamente, oltre all'informazione scritta, l'immagine fotografica. L'esposizione relativa alla gestione del video e alla grafica comprende anche la descrizione delle mappe di memoria del video, passo fondamentale per entrare nelle problematiche della «computer-grafica».
Tra le molteplici attitudini e possibilità del Commodore 64 è inclusa anche la capacità di emettere una vasta gamma di note musicali, suoni e rumori. Che un computer sia in grado di suonare ormai non stupisce più nessuno, ma si deve essere ben consci del fatto che anche il computer più sofisticato, come del resto qualsiasi strumento musicale, non può suonare «da solo»: è necessario che esegua un programma appositamente preparato.
L'ultimo paragrafo della parte descrittiva dell'hardware è dedicato dunque al suono, e consentirà di acquisire tutti gli elementi necessari per utilizzare anche questa attitudine del Commodore 64 nelle nostre esperienze di «computer-music».
La prima parte del manuale si conclude con un'esposizione completa ed esauriente, ma soprattutto facilmente comprensibile, del linguaggio Basic utilizzato dal Commodore 64. Sono state distinte funzionalmente e in modo progressivo le descrizioni delle variabili, degli operatori, dei comandi, delle istruzioni e delle funzioni del Basic. Al di là delle tendenze

più tradizionali, spesso improntate ad un freddo schematismo, si sono adottate una classificazione ed una sequenzialità specificamente orientate al programmatore, con l'intento di facilitare la comprensione del linguaggio e di costituire un utile strumento di consultazione.
La seconda e principale parte del manuale è dedicata invece allo sviluppo delle metodologie proprie della programmazione attraverso l'impostazione e la soluzione logica di venti problemi concreti.
Lo sviluppo di un programma si articola in tre fasi distinte: l'analisi del problema che si vuole risolvere, l'organizzazione logica nei diagrammi di flusso del metodo di soluzione prescelto, la minutazione del programma nel linguaggio di programmazione che si intende adottare. Nelle venti applicazioni che presentiamo, ogni singola fase è affrontata separatamente in maniera specifica e dettagliata, così da fornire al lettore un quadro metodologico completo del procedimento da seguire.

Analisi del problema

In questa fase sono anzitutto enumerati i requisiti del programma che si intende realizzare. In base ad essi sono successivamente individuati e descritti i metodi (algoritmi) di soluzione, adottando una forma di esposizione per quanto possibile chiara e discorsiva.

Diagrammi di flusso

L'algoritmo di soluzione è organizzato in maniera logica e sequenziale, seguendo per quanto possibile i canoni della programmazione strutturata. Ogni diagramma di flusso è puntualmente commentato a margine, per meglio chiarire il significato delle soluzioni adottate e le funzionalità dell'algoritmo.

Il programma

Questa parte riguarda la traduzione dei diagrammi di flusso nelle istruzioni previste dal Basic, riportando per esteso i listati completi dei programmi e inserendo un duplice livello di commento: un frequente inserimento nei listati delle istruzioni di commento e una descrizione a margine, discorsiva e puntuale, delle singole istruzioni utilizzate.

Non è forse inutile sottolineare, prima di concludere, il carattere prevalentemente didattico di questo manuale, che tuttavia, anziché perdersi dietro tediose e (perché no) imbarazzanti dissertazioni sui microcircuiti, può diventare il tramite amichevole e divertente fra una persona e una macchina.

single drive
floppy disk
VC-1541

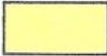
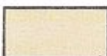
Il personal computer Commodore 64

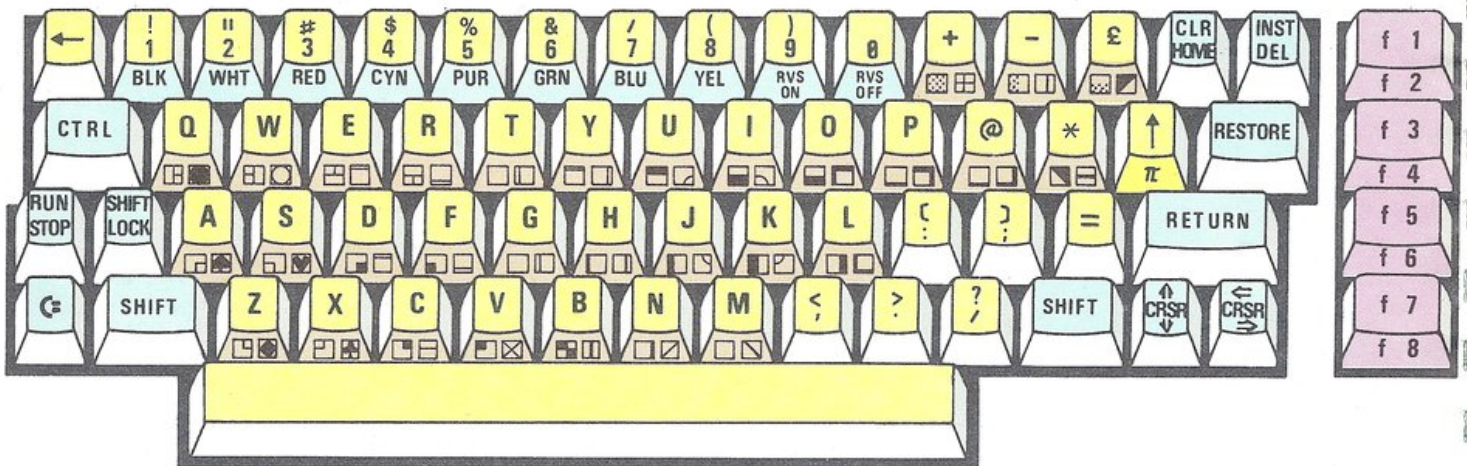
Così come il suo fratello minore VIC-20, il Commodore 64, che d'ora in poi chiameremo più semplicemente CBM-64, è un home computer, un calcolatore destinato ad applicazioni «domestiche», cioè di limitata complessità. Il CBM-64 è composto da una unità fisica comprendente:

- la **tastiera**, utilizzata per il dialogo con la macchina
- la **CPU** (Central Processing Unit, unità centrale di elaborazione), la parte della macchina che esegue fisicamente i programmi
- la **memoria**, divisa in **ROM** (Read Only Memory), contenente il programma di sistema che gestisce le operazioni eseguite dalla macchina, e in **RAM** (Random Access Memory), usata per i programmi scritti dagli utenti. La memoria del CBM-64 ha una capacità di 64 kbytes.

Il CBM-64 può essere connesso o ad un apposito monitor oppure, attraverso un modulatore a radiofrequenza incluso nello chassis della tastiera, ad un comune televisore a colori, che può sostituire egregiamente il monitor del calcolatore, anche se con una perdita di qualità dell'immagine. Il computer utilizza questo componente per comunicare messaggi, per evidenziare i risultati dell'esecuzione di un programma e, in generale, per visualizzare tutto quanto viene digitato dall'operatore sulla tastiera. In quest'ultimo caso si dice che sul video compare l'«eco» di quanto viene digitato.
Questa macchina può essere inoltre dotata di memorie non volatili a supporto magnetico, come **cassette** o **floppy-disks**; le cassette sono utilizzabili mediante un apposito registratore **Datassette**, mentre i floppy-disks richiedono una diversa e più complessa apparecchiatura detta **driver**.
I due tipi di supporto magnetico differiscono essenzialmente per la diversa velocità con cui si possono scrivere o leggere le informazioni (la gestione del dischetto è molto più veloce di quella della cassetta).
Allo stesso scopo sono adottate cartucce di espansione di memoria ROM contenenti programmi di grosse dimensioni che potrebbero essere caricati con difficoltà nella memoria della configurazione base. Le cartucce di espansione vanno inserite nello speciale slot (alloggiamento) appositamente previsto sul retro della tastiera.
Al fine di ovviare alla necessità di avere copie su carta di quanto caricato nella macchina, il CBM-64 è dotato di un dispositivo (**bus seriale**) al quale è possibile collegare una o più stampanti, e dispone inoltre di una presa (interfaccia IEEE-488) che consente il collegamento con altre apparecchiature di impiego particolare (**plotter**, cioè tracciatori di grafici, strumenti di laboratorio, ecc.).
Le potenzialità del CBM-64 possono essere ulteriormente espanse collegandolo con altri computer più potenti. Tale collegamento può essere stabilito attraverso le normali linee telefoniche, ma è necessario in tal caso inserire tra il computer e la rete telefonica un dispositivo chiamato **modem** (modulatore-demodulatore), che elabora i segnali digitali in uscita dal CBM-64 per renderli idonei alla trasmissione sulla linea telefonica e viceversa.
Il CBM-64 utilizza la forma di trasmissione più semplice e più diffusa, denominata protocollo RS-232, e pertanto può essere collegato alla maggior parte dei calcolatori esistenti.
Infine è possibile corredare il CBM-64 di dispositivi di comando, come i **joystiks**, i **paddles** e le **penne ottiche**, che trovano impiego nei videogiochi e in altre applicazioni più impegnative.

La tastiera

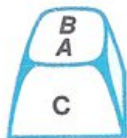
La tastiera del CBM-64 è analoga a quella di una normale macchina per scrivere.
I tasti si possono suddividere in quattro categorie:

- **tasti di scrittura** (lettere, numeri, caratteri speciali)
- **tasti grafici**
- **tasti di controllo**, utilizzati per inviare alcuni comandi al CBM-64
- **tasti funzionali**, che possono attivare determinate azioni all'interno di programmi scritti dall'utente.

I diversi caratteri rappresentati su un tasto possono essere selezionati per mezzo dei tasti SHIFT e COMMODORE (C=).
Bisogna chiarire subito che uno stesso tasto (fisico) può appartenere ad una o a più di una delle categorie enumerate. Ad esempio, quasi tutti i tasti che producono caratteri di scrittura recano impressi sulla faccia anteriore anche due caratteri grafici, che possono essere attivati premendo quel tasto in concomitanza con un opportuno tasto di controllo.

Tasti di scrittura — producono caratteri di scrittura, cioè lettere, numeri e caratteri speciali

Tasti grafici — producono caratteri grafici, utilizzabili per comporre disegni

Tasti di controllo — modificano il funzionamento della macchina

Tasti funzionali — possono attivare, all'interno di una data elaborazione, particolari azioni che l'utente può programmare

Tasti di scrittura

I tasti di scrittura hanno un funzionamento molto simile a quello dei tasti di una macchina per scrivere.
La semplice pressione di un tasto di scrittura produce la comparsa sul video, nella posizione occupata dal cursore, del carattere associato al tasto premuto. Contemporaneamente il cursore si sposta di un posto a destra, segnando il punto in cui avverrà la prossima immissione.
I tasti di scrittura occupano la zona della tastiera evidenziata in colore nell'illustrazione a margine. Nella quasi totalità dei casi i caratteri di scrittura sono impressi sulla faccia superiore del tasto, ad eccezione di quello che attiva il carattere π, che reca impresso il carattere sulla faccia anteriore. In generale, l'operazione che consente di ottenere sul video un carattere di scrittura è diversa a seconda della posizione che occupa sul tasto il carattere desiderato.

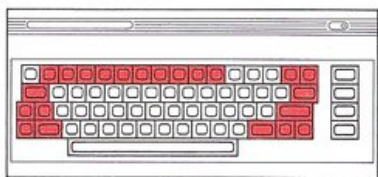
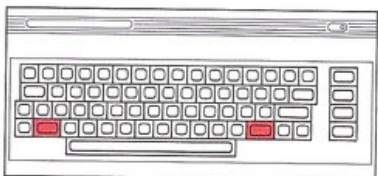
- Per ottenere il carattere in posizione **A** è sufficiente premere semplicemente il tasto. In questo modo si possono digitare, ad esempio, i caratteri A, B, C, ecc.
- Per ottenere il carattere in posizione **B** è necessario tenere premuto il tasto SHIFT e contemporaneamente premere quello desiderato. In questo modo si possono avere, ad esempio, i caratteri ! " # ecc.
- Per ottenere il carattere in posizione **C** è necessario tenere premuto il tasto SHIFT e premere contemporaneamente quello che reca il carattere. L'unico carattere di scrittura che può essere ottenuto in questo modo è il pi greco (π).

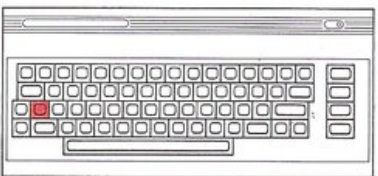
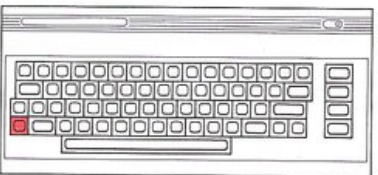
La prima cosa che si nota digitando i caratteri di scrittura è che il CBM-64 visualizza le lettere dell'alfabeto in maiuscolo (stato maiuscole/grafici). Questa caratteristica è necessaria perché molti linguaggi di programmazione prevedono l'uso delle lettere maiuscole per i comandi e le istruzioni. Il CBM-64 può tuttavia entrare in un modo di funzionamento particolare, nel quale è possibile utilizzare maiuscole e minuscole. Per entrare nello stato maiuscole/minuscole (chiamato anche **stato testo,** proprio perché più adatto alla composizione dei testi) è sufficiente tenere premuto il tasto SHIFT e premere contemporaneamente il tasto COMMODORE. Una volta entrati in questo stato, i caratteri di scrittura digitati sono visualizzati in minuscolo, a meno che non venga premuto contemporaneamente alla digitazione del tasto-carattere il tasto SHIFT, proprio come accade in una macchina per scrivere.
Per uscire dallo stato testo è sufficiente ripetere la stessa operazione eseguita per entrarvi.

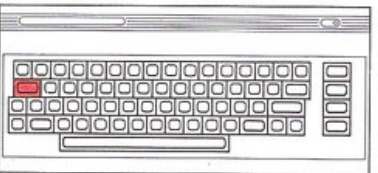
Tasti grafici

Una caratteristica molto importante del CBM-64 è l'insieme dei caratteri grafici previsti sulla tastiera, utilizzabili per comporre sul video grafici, disegni e tabelle.
I caratteri grafici sono attivati dai tasti che occupano la zona evidenziata in colore nell'illustrazione a margine.
Ciascuno dei tasti indicati reca impressi sulla faccia anteriore due caratteri grafici, che nella riproduzione illustrativa riportata qui a margine abbiamo sostituito, a titolo di esempio, con le lettere **A** e **B,** facendo così riferimento solo alla loro posizione sul tasto.

- Per attivare il carattere che occupa la posizione **A** è necessario tenere premuto il tasto COMMODORE e premere contemporaneamente quello desiderato.
- Per ottenere il carattere che occupa la posizione **B** è necessario tenere premuto il tasto SHIFT e premere contemporaneamente quello desiderato.

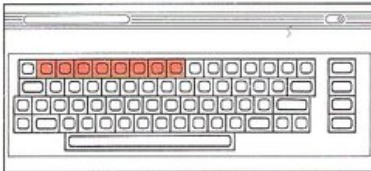
Tasti di controllo

La tastiera del CBM-64, come quella di qualsiasi altro computer, include un certo numero di tasti che, se premuti, non attivano caratteri di scrittura o grafici.
Questi tasti, chiamati tasti di controllo, generano all'atto della pressione un segnale (di controllo, appunto), che in genere chiede alla macchina di modificare il suo funzionamento.
Si tratta, in altri termini, di un insieme di tasti la cui pressione consente all'operatore di «governare» la macchina.

I tasti **SHIFT**, in numero di due, uno a sinistra e l'altro a destra, coprono diverse funzioni. La pressione del tasto SHIFT consente:

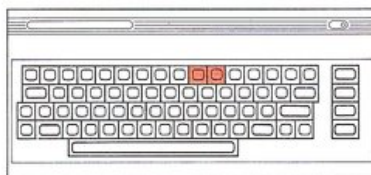
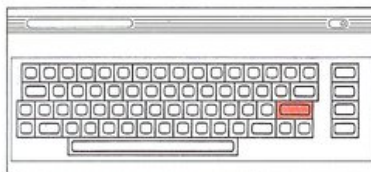
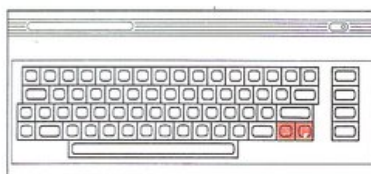
- di attivare il carattere π
- di ottenere i caratteri di scrittura impressi in alto sulla parte superiore dei tasti (p. es. ! $ " ecc.)
- di ottenere i caratteri grafici impressi a destra sulla superficie anteriore dei tasti
- in concomitanza con la pressione del tasto COMMODORE, di entrare nello stato testo e di uscirne
- di ottenere le lettere maiuscole quando si opera in modalità testo
- di selezionare una delle due funzioni proprie di ciascun tasto funzionale
- di selezionare una delle due funzioni proprie di ciascun tasto di controllo.

Il tasto **SHIFT LOCK** serve per conservare indefinitamente la situazione che si determina quando è premuto il tasto SHIFT.
Può infatti capitare di dover digitare una lunga sequenza di caratteri che richiedono la contemporanea pressione del tasto SHIFT: ad esempio, i caratteri grafici impressi a destra sulla superficie anteriore dei tasti. In questi casi, si può premere lo SHIFT LOCK, che agisce come un interruttore a bottone: fin quando non si «spegne» questo interruttore è come se il tasto SHIFT fosse premuto.

Il tasto **COMMODORE**, che è una caratteristica specifica del CBM-64 e delle macchine Commodore, svolge tre funzioni di selezione:

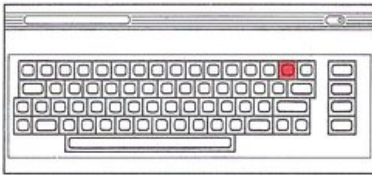
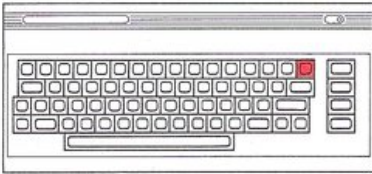
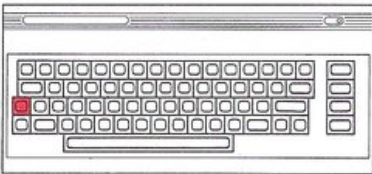
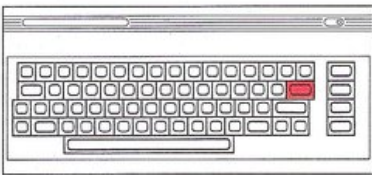
- consente di attivare i caratteri grafici che si trovano a sinistra sulla superficie anteriore dei tasti (successione COMMODORE+tasto)
- consente di passare nel modo di funzionamento maiuscole/minuscole (successione COMMODORE+SHIFT) e viceversa
- consente di selezionare la seconda serie di 8 colori: tenendo premuto il tasto COMMODORE e uno dei tasti di selezione colore si ottiene l'uso di altre tonalità.

Il tasto **CTRL** svolge le seguenti funzioni:

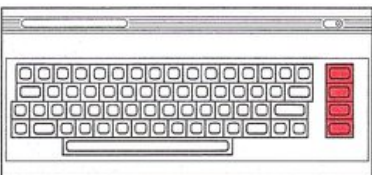
- consente di attivare i tasti di controllo del colore in scrittura sul video. In condizioni normali, il CBM-64 visualizza i caratteri in azzurro su fondo blu. Se l'operatore, tenendo premuto il tasto CTRL, preme uno dei tasti di controllo del colore, a partire da questo momento i caratteri che verranno digitati saranno visualizzati, anziché in azzurro, nel colore selezionato
- consente di attivare i tasti REVERSE (RVS ON e RVS OFF) che attivano e disattivano la visualizzazione invertita. Se l'operatore, tenendo premuto il tasto CTRL, preme il tasto RVS ON, attiva la visualizzazione invertita: a partire da questo momento tutti i caratteri che saranno digitati appariranno scritti in blu su un rettangolino azzurro
- consente di rallentare l'esecuzione dei programmi: se si tiene premu-

to il tasto CTRL mentre il CBM-64 sta eseguendo un programma, la velocità di esecuzione diminuisce

- consente all'operatore di definire comandi di controllo propri, che possono essere inclusi in una qualsiasi applicazione
- è utilizzato da alcuni programmi caricati su cartucce ROM per svolgere funzioni particolari.

I tasti di **controllo del colore** sono in numero di otto, e si trovano nella prima fila della tastiera. Si tratta degli stessi tasti che, con la semplice pressione, attivano la visualizzazione dei numeri da 1 a 8.
I segnali di controllo del colore sono invece attivati premendo questi tasti in concomitanza con il tasto CTRL o con il tasto COMMODORE: ciò permette di selezionare 16 colori. Se, tenendo premuto uno di questi due tasti, si preme uno dei tasti di controllo colore, a partire da quel momento tutti i caratteri digitati sulla tastiera appariranno sul video nel colore selezionato.
In particolare:

premendo i tasti	i caratteri appariranno	premendo i tasti	i caratteri appariranno
CTRL + BLK	neri	C= + BLK	arancioni
CTRL + WHT	bianchi	C= + WHT	marroni
CTRL + RED	rossi	C= + RED	rosso chiaro
CTRL + CYN	blu-verdi	C= + CYN	grigio scuro
CTRL + PUR	porpora	C= + PUR	grigio
CTRL + GRN	verdi	C= + GRN	verde chiaro
CTRL + BLU	blu	C= + BLU	azzurri
CTRL + YEL	gialli	C= + YEL	grigio chiaro

I tasti **RVS ON** e **RVS OFF** attivano e disattivano la funzione REVERSE, che determina la visualizzazione dei caratteri «in negativo». Se l'operatore tiene premuto il tasto CTRL e preme contemporaneamente il tasto RVS ON, a partire da questo momento i caratteri digitati appariranno sullo schermo in blu su fondo azzurro. Questa modalità di funzionamento proseguirà fin quando, premendo contemporaneamente il tasto CTRL e il tasto RVS OFF, non verrà disattivata la funzione REVERSE.

Il tasto **RETURN** consente di restituire alla macchina il controllo che l'operatore assume quando digita una parola, una riga, una frase o un'istruzione. Restituire il controllo significa consentire alla macchina di elaborare quanto l'operatore ha digitato sulla tastiera. È necessario premere il tasto RETURN dopo aver digitato ciascuna istruzione di un programma. Solo all'atto dell'attivazione del tasto il CBM-64 può leggere, riconoscere, elaborare ed eventualmente eseguire l'istruzione, che altrimenti è solo visualizzata sullo schermo.

I due tasti **CRSR** servono per spostare il cursore (CURSOR) all'interno della zona del video nella quale è consentito l'inserimento dei caratteri. Un qualsiasi carattere che venga digitato sulla tastiera va sempre ad occupare la posizione indicata dal cursore. Se si vuole modificare la posizione d'inserimento rispetto a quella «puntata» dal cursore è quindi necessario spostare quest'ultimo nella posizione desiderata. Il primo tasto consente lo spostamento verticale del cursore, il secondo quello orizzontale. La direzione dello spostamento (verso l'alto o verso il basso, verso sinistra o verso destra) è indicata dalle frecce. Per effettuare uno spostamento verso una direzione indicata dalla freccia che si trova sotto la si-

gla CRSR, è sufficiente la semplice pressione del tasto; per effettuare invece uno spostamento nella direzione indicata dalla freccia che si trova sopra la sigla CRSR, è necessario premere contemporaneamente il tasto SHIFT. Le possibilità di spostamento del cursore sono preziose quando si deve correggere una parola, un'intera frase o un'istruzione visualizzata. È sufficiente posizionare il cursore sopra i caratteri errati e digitare quelli giusti.

Al tasto **CLR/HOME** sono associate due diverse funzioni, contrassegnate dalle sigle CLR (contrazione di CLEAR) e HOME.
La funzione HOME (casa) si attiva alla semplice pressione del tasto e produce lo spostamento istantaneo del cursore nella prima casella in alto a sinistra sul video (la «casa» del cursore). Ciò che è visualizzato non viene cancellato. Per attivare invece la funzione CLEAR (cancellazione) è necessario premere contemporaneamente il tasto SHIFT. Il suo effetto è quello di cancellare tutto ciò che appare sul video, riportando il cursore nella prima casella in alto a sinistra.

Al tasto **INST/DEL** sono associate due diverse funzioni (INSERT, inserisci, e DELETE, cancella). La funzione DEL è attivata dalla semplice pressione del tasto, e causa la cancellazione sul video del primo carattere alla sinistra del cursore, con l'eliminazione automatica dello spazio che ne risulta. La funzione INSERT causa invece l'inserimento di uno spazio in bianco nella posizione del cursore. Si attiva tenendo premuto il tasto SHIFT e premendo contemporaneamente il tasto INST/DEL.

Il tasto **RUN/STOP** attiva la funzione di INTERRUPT (interruzione), ordina cioè al CBM-64 di interrompere ciò che sta facendo e di restituire il controllo all'operatore.
A questo punto, il CBM-64, attraverso il video, invia all'operatore un messaggio per avvertirlo che l'esecuzione è stata interrotta, specificando inoltre a che punto del programma (numero dell'istruzione) si è verificata l'interruzione. Sul video ricompare il cursore, che non è presente in fase di esecuzione.
La pressione del tasto RUN/STOP in concomitanza con lo SHIFT attiva il codice LOAD (carica), che ordina al CBM-64 di iniziare il caricamento in memoria di un programma registrato su nastro.

Se si tiene premuto il tasto RUN/STOP e contemporaneamente si preme il **RESTORE**, si invia al CBM-64 un segnale che attiva il ripristino del computer nella condizione che esisteva immediatamente dopo l'accensione, con il vantaggio che eventuali programmi caricati in memoria non vengono persi. Il codice RESTORE genera in altri termini un'operazione di pulizia della memoria che consente di rieseguire da capo un dato programma, o di listarlo.

Tasti funzionali

I tasti funzionali, chiamati anche **tasti di funzione programmabili**, sono presenti sulla tastiera del CBM-64 in numero di quattro all'estrema destra. A ciascuno di essi possono essere associate, mediante apposito programma, due diverse funzioni: la prima sarà attivata alla semplice pressione di quel tasto, mentre per attivare la seconda sarà necessario premere contemporaneamente anche lo SHIFT. Le funzioni attivabili nel primo modo sono quelle poste sulla faccia superiore del tasto (f1, f3, f5, f7); nel secondo modo sono attivate quelle poste sulla faccia anteriore (f2, f4, f6, f8).
I tasti funzionali sono utilizzati anche da molti programmi caricati sulle cartucce ROM ad innesto per consentire all'utente di effettuare particolari scelte.

Il video

Il CBM-64 è in grado di gestire un'immagine video a colori in cui possono entrare 25 righe di 40 caratteri ciascuna. Si dice anche, con maggior rigore, che si tratta di un video a 25 righe per 40 colonne.
Il video è utilizzato come periferica di uscita (output) standard: ciò significa che, per rispondere a comandi e istruzioni, il CBM-64 utilizzerà sempre il video, a meno di non chiedergli esplicitamente, tramite appositi comandi, di utilizzare altre periferiche (per esempio, la stampante).
All'atto dell'accensione del computer appare sul video un riquadro blu (la zona in cui è possibile inserire caratteri) contornato da un margine azzurro. In alto appare il messaggio "* * * * COMMODORE 64 BASIC V2 * * * * 64K RAM SYSTEM 38911 BASIC BYTES FREE", scritto in azzurro nel riquadro blu, che dà all'operatore alcune informazioni circa il linguaggio che deve utilizzare e la capacità della memoria RAM di cui dispone. Subito sotto si può scorgere il messaggio READY e il cursore.
Se vogliamo iniziare a digitare qualcosa possiamo ordinare al CBM-64 di cancellare il video attivando la sequenza di tasti SHIFT + CLR/HOME.

Il cursore

Il cursore appare ora come un rettangolo lampeggiante di colore azzurro che occupa la prima casella in alto a sinistra (home). Esso indica, in ogni momento, la posizione in cui sarà inserito il primo carattere che verrà digitato. Ad ogni immissione il cursore si sposterà di una posizione verso destra, fino a raggiungere il margine destro della zona blu, quindi tornerà automaticamente a capo della riga successiva, pronto ad accettare nuove immissioni.
La posizione di inserimento dei caratteri può essere modificata a piacere spostando il cursore mediante gli appositi tasti di controllo (CRSR).
Dobbiamo solo notare che, se premiamo uno dei tasti di controllo del cursore dopo aver digitato un doppio apice, il CBM-64 evidenzia uno dei caratteri mostrati nella foto sotto.

La risposta fornita dal CBM-64 quando si impiegano i tasti di controllo del cursore fra apici, ad esempio nelle istruzioni PRINT.

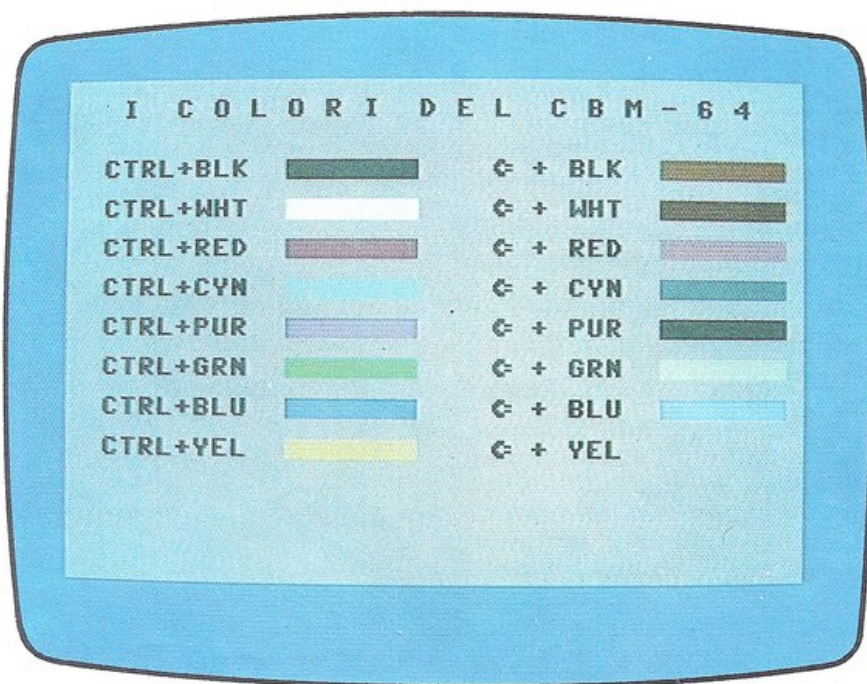
L'impiego dei tasti di controllo del cursore fra apici si presenta allorché si desidera spostare il cursore durante l'esecuzione di un programma con un'istruzione PRINT.

Il colore

Illustrando le funzioni dei tasti di controllo del CBM-64, si è già parlato della possibilità di visualizzare i caratteri digitati in sedici colori diversi, in positivo o invertiti; ma le possibilità cromatiche del calcolatore non si fermano alle funzioni RVS e ai tasti di controllo del colorè.
È infatti possibile variare i colori del margine del video e del riquadro che delimita la zona riservata alle immissioni: il calcolatore può adottare per il margine e per lo schermo sedici colori diversi. I cambiamenti di colore si attivano immettendo due appositi comandi, che hanno la seguente sintassi:

POKE 53280,X	per il colore del margine
POKE 53281,Y	per il colore del riquadro

dove X e Y possono assumere diversi valori.
Il significato del comando e il suo funzionamento richiedono qualche precisazione sulla struttura della memoria del CBM-64.
Si può immaginare quest'ultima come un grande tabellone costituito da tante celle numerate (bytes). La presenza di un certo valore in una determinata cella ha per il calcolatore un preciso significato, che dipende dal valore che è stato immesso in quella cella.
Con i comandi POKE 53280,X e POKE 53281,Y si ordina al CBM-64 di inserire nelle rispettive celle di memoria i valori numerici di X ed Y.
Quando il CBM-64 è acceso, un programma di sistema legge regolarmente il contenuto delle celle di memoria preposte al controllo delle diverse funzioni e, a seconda del valore numerico che trova nelle celle 53280 e 53281, attiva la corrispondente combinazione di colori.
I valori numerici da inserire al posto delle X e delle Y per avere le varie combinazioni di colore sono elencati nella tabella riportata a fianco.

0	NERO
1	BIANCO
2	ROSSO
3	BLU-VERDE
4	PORPORA
5	VERDE
6	BLU
7	GIALLO
8	ARANCIONE
9	MARRONE
10	ROSSO CHIARO
11	GRIGIO SCURO
12	GRIGIO
13	VERDE CHIARO
14	AZZURRO
15	GRIGIO CHIARO

I colori che il CBM-64 può utilizzare per scrivere i caratteri. A fianco sono elencati i tasti di controllo da premere per selezionarli.

Proviamo ora a variare i colori del nostro video in modo da ottenere margine e schermo neri, con i caratteri visualizzati in bianco. Dovremo allora introdurre la sequenza di comandi

POKE 53280,0 RETURN
POKE 53281,0 RETURN

A questo punto lo schermo è tutto nero, con il cursore in azzurro. Non ci rimane che digitare

CTRL+WHT

e il cursore diventa bianco: il nostro video ha assunto ora un'aria molto professionale.
Per tornare alla normale combinazione di colori è necessario immettere la seguente sequenza di comandi

POKE 53280,14 RETURN
POKE 53281,6 RETURN
C= +BLU

L'impiego del colore nei programmi

Per poter riprodurre su video un'immagine a colori senza doverla ricomporre ogni volta da capo è necessario scrivere un programma che produca la visualizzazione dei caratteri grafici utilizzati nel giusto colore e nella giusta posizione. Programmi di questo genere sono di solito molto semplici ed utilizzano una sola istruzione: la PRINT.
Scrivendo l'istruzione

1 PRINT "ABCD..." RETURN

abbiamo in realtà immesso un programma che causa la visualizzazione (PRINT) sul video di tutto ciò che è compreso fra i doppi apici. Il numero 1 serve al CBM-64, in presenza di più istruzioni, per capire quale di esse deve eseguire per prima.
Per avere la scritta ABCD... su video occorre ora mandare in esecuzione il programma, e ciò si ottiene inserendo il comando

RUN RETURN

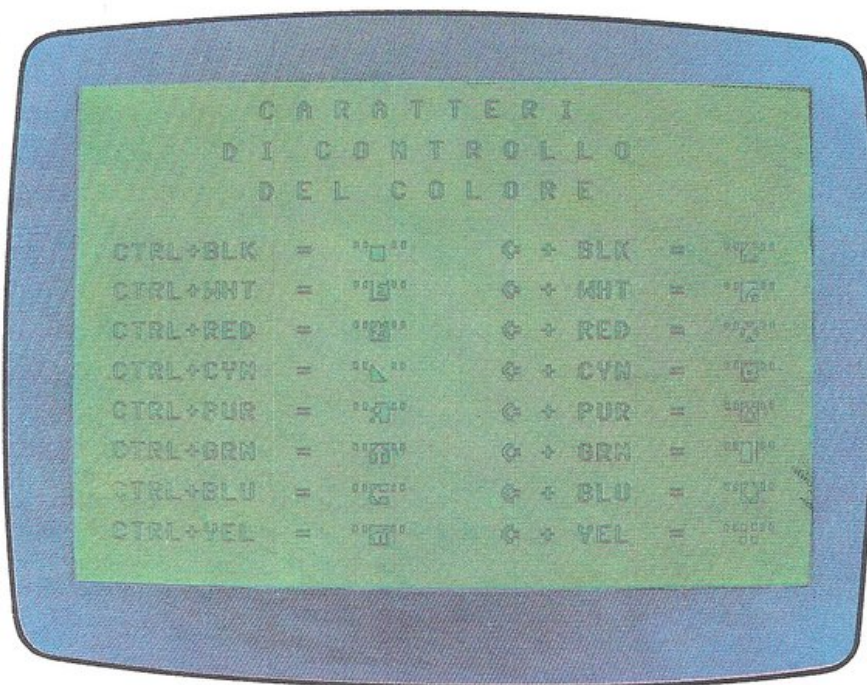
Appena premuto il tasto RETURN, dopo aver digitato la parola RUN, il calcolatore esegue il programma, e sul video appaiono la scritta che avevamo incluso nell'istruzione 1 e il messaggio READY, col quale il CBM-64 ci avverte che ha eseguito il nostro comando RUN.
Se si vuole stampare una sequenza di caratteri (di scrittura o grafici) in un colore diverso da quello che il CBM-64 utilizza abitualmente, è necessario inserire nell'istruzione di stampa, prima del carattere, il codice di controllo del colore che si vuole utilizzare.
Ad esempio, se si vuole scrivere la sequenza ABCD... in rosso occorre digitare

1 PRINT "CTRL+RED ABCD..." RETURN

L'eco della sequenza dei tasti di controllo CTRL+RED che attivano la visualizzazione in rosso è rappresentata sul video col simbolo £ (simbolo della sterlina invertito), per cui ciò che vedremo comparire è

1 PRINT "£ ABCD..."

Mandando in esecuzione questo programma (comando RUN) la scritta ABCD... sarà visualizzata dal CBM-64 in rosso.
Quando è utilizzato fra i doppi apici, cioè all'interno di un'istruzione PRINT, ogni comando di controllo del colore produce sul video un'eco diversa, come si può verificare nell'immagine video riportata qui sotto.

La risposta fornita dal CBM-64 all'impiego fra apici dei tasti di controllo del colore.

Le mappe di memoria del video

L'immagine video è gestita dal CBM-64 grazie a due **mappe di memoria**, l'una preposta alla memorizzazione dei codici dei caratteri da visualizzare, l'altra a quella dei codici dei colori che detti caratteri devono avere. Ciò significa che il calcolatore deve tenere costantemente aggiornate in memoria le due tabelle riportate nella pagina a fianco.
Le due mappe hanno dimensioni $25 \times 40 = 1000$ bytes ciascuna (uno per ogni posizione sul video). La mappa dei colori è memorizzata a partire dalla locazione 55296, mentre la mappa di memoria dei caratteri inizia alla locazione 1024. I numeri indicati consentono di ricavare gli indirizzi delle celle di memoria che contengono, per ciascuna posizione, il codice del carattere che vi deve comparire (TAB. 1) e il codice del colore (TAB. 2). I numeri a sinistra indicano gli indirizzi delle celle che controllano la prima colonna del video; gli indirizzi delle celle che controllano una delle caselle interne si possono ottenere sommando i numeri della riga e della colonna di cui la casella interna in questione fa parte. Tanto per fare un esempio, quanto è visualizzato nella prima casella in alto a sinistra del video dipende:

- dal contenuto della locazione di memoria 1024 per quanto riguarda il carattere visualizzato
- dal contenuto della locazione di memoria 55296 per quanto riguarda il colore del carattere visualizzato.

Le celle di memoria possono contenere solo numeri, perciò è chiaro che caratteri e colori devono essere memorizzati sotto forma di codici numerici. Il valore numerico da inserire in una data cella di memoria per ottenere nella posizione corrispondente del video un determinato ca-

TAB. 1 MAPPA DI MEMORIA DEI CARATTERI

COLONNE

TAB. 2 MAPPA DI MEMORIA DEI COLORI

COLONNE

rattere si può desumere dalla tabella di pag. 18, dove sono indicati

- i caratteri rappresentabili in modalità maiuscole/grafici (serie 1)
- i caratteri rappresentabili in modalità testo (serie 2)
- i corrispondenti codici numerici (POKE).

I valori numerici che selezionano i diversi colori sono riportati nella tabella di pag. 14. La conoscenza delle mappe di memoria del video consente di comporre grafici e disegni assegnando direttamente il contenuto di ciascuna casella del video con l'istruzione POKE. Se ad esempio vogliamo visualizzare nella prima posizione in alto a sinistra del video una pallina (codice 81 della tabella dei caratteri) di colore rosso (codice 2 della tabella dei colori), dovremo immettere i seguenti comandi:

POKE 55296,2	attiva il colore rosso
POKE 1024,81	attiva la visualizzazione del carattere «pallina»

SERIE1	SERIE2	POKE	SERIE1	SERIE2	POKE	SERIE1	SERIE2	POKE	SERIE1	SERIE2	POKE	SERIE1	SERIE2	POKE	SERIE1	SERIE2	POKE
@		0	U	u	21	*		42	?		63		T	84			106
A	a	1	V	v	22	+		43			64		U	85			107
B	b	2	W	w	23	,		44	♠	A	65	╳	V	86			108
C	c	3	X	x	24	—		45		B	66	○	W	87			109
D	d	4	Y	y	25	.		46		C	67	♣	X	88			110
E	e	5	Z	z	26	/		47		D	68		Y	89			111
F	f	6	[		27	Ø		48		E	69	♦	Z	90			112
G	g	7	£		28	1		49		F	70			91			113
H	h	8	]		29	2		50		G	71			92			114
I	i	9	↑		30	3		51		H	72			93			115
J	j	10	←		31	4		52		I	73	π		94			116
K	k	11	SPACE		32	5		53		J	74			95			117
L	l	12	!		33	6		54		K	75	SPACE		96			118
M	m	13	"		34	7		55		L	76			97			119
N	n	14	#		35	8		56		M	77			98			120
O	o	15	$		36	9		57		N	78			99			121
P	p	16	%		37	:		58		O	79			100		✓	122
Q	q	17	&		38	;		59		P	80			101			123
R	r	18	'		39	<		60	●	Q	81			102			124
S	s	19	(		40	=		61		R	82			103			125
T	t	20	)		41	>		62	♥	S	83			104			126
														105			127

I codici da 128 a 255 sono le immagini in negativo dei codici da 0 a 127.

La grafica

Il CBM-64 gestisce l'insieme di 256 caratteri standard riprodotto qui sopra. La configurazione grafica di ogni singolo carattere è immagazzinata nella memoria ROM a partire da un indirizzo di base, memorizzato nel registro 53272. Ad ogni carattere è associato un insieme di 8 bytes consecutivi che ne definiscono la rappresentazione sul video. Ad esempio, la rappresentazione in memoria ROM del carattere B sarà definita come indicato nella figura a margine della pagina a fianco: ad ogni bit posto a 1 corrisponderà un punto scuro sul video, mentre ai bits posti a 0 corrisponderà un punto chiaro.
Il CBM-64 offre all'utente la possibilità di ridefinire graficamente i caratteri all'interno della memoria accessibile (RAM), in modo da ottenere un insieme personalizzato. Per fare questo è necessario trasferire la

bit

byte	7	6	5	4	3	2	1	0
0	0	1	1	1	1	1	0	0
1	0	0	1	0	0	0	1	0
2	0	0	1	0	0	0	1	0
3	0	0	1	1	1	1	0	0
4	0	0	1	0	0	0	1	0
5	0	0	1	0	0	0	1	0
6	0	1	1	1	1	1	0	0
7	0	0	0	0	0	0	0	0

rappresentazione dei caratteri standard che si vogliono conservare dalla ROM alla RAM, quindi definire all'interno di quest'ultima la forma dei nuovi caratteri e poi comunicare al sistema che deve prelevare le informazioni relative ai caratteri da lì, e non più dalla ROM. Una precauzione da prendere prima di definire il nuovo insieme di caratteri consiste nel proteggere la zona di memoria che si vuole riservare per questa operazione dalla sovrapposizione del programma Basic, in quanto anch'esso usa la memoria RAM. Questo si ottiene con l'istruzione

```
POKE 52, 48: POKE 56, 48: CLR
```

Come si è detto, il CBM-64 è provvisto di un registro programmabile (indirizzo 53272) che punta alla zona di memoria contenente le definizioni dei caratteri; è quindi sufficiente variare il contenuto di tale registro mediante l'istruzione POKE, affinché il sistema possa prelevare la nuova rappresentazione dei caratteri dalla zona di memoria RAM il cui indirizzo di partenza sarà specificato nella POKE stessa.
Il punto di partenza più comodo in cui porre l'insieme dei caratteri personalizzati è a partire dalla locazione 12288.
La nuova rappresentazione di un carattere può essere definita scrivendo dapprima una nuova matrice 8x8 che rappresenti la nuova forma del carattere sullo schermo, caricando poi questa matrice nelle locazioni che descrivono il carattere in questione.
Immaginiamo ad esempio di voler costruire un insieme di caratteri personalizzato contenente i primi 64 caratteri standard, in cui la rappresentazione grafica del carattere T sia sostituita con un omino.
Cominceremo con l'impostare le maiuscole e col riservare la zona di memoria per il nuovo insieme di caratteri tramite le istruzioni

```
5 PRINT CHR$(142)
10 POKE 52, 48: POKE 56, 48: CLR
```

Sarà poi necessario disattivare le funzioni di I/O, per permettere di posizionare la ROM caratteri a partire dalla locazione 53248; queste due operazioni vengono eseguite in sequenza tramite le istruzioni

```
20 POKE 56334,PEEK(56334) AND 254
30 POKE 1,PEEK (1) AND 251
```

Trasferiremo ora i primi 64 caratteri standard di scrittura (maiuscoli) nella memoria RAM a partire dalla locazione 12288 con la seguente istruzione:

```
40 FOR I=0 TO 511: POKE I+12288,PEEK (I+53248):NEXT I
```

Così facendo preleveremo i bytes memorizzati nelle locazioni da 53248 a 53759 per copiarli in quelle da 12288 a 12799. Ora dovremo riposizionare la ROM di I/O (a partire dalla locazione 53248) e riattivare la tastiera con i seguenti comandi

```
50 POKE (1),PEEK(1) OR 4
60 POKE 56334,PEEK (56334) OR 1
```

Dovremo poi indicare al sistema che deve prelevare la rappresentazione dei caratteri a partire dalla locazione di memoria 12288, modificando i 4 bits meno significativi della locazione 53272 (registro puntatore) con il valore 12 (indirizzo relativo alla locazione 12288), mediante l'istruzione

```
70 POKE 53272, (PEEK (53272) AND 240)+12
```

A questo punto definiremo il nuovo carattere (omino) associandolo al carattere T. Questo carattere si trova ora in 8 bytes consecutivi a partire dalla locazione 12288+(8x20)=12448 (essendo T il carattere numero 20). In ciascuno di questi bytes dovremo trasferire il valore binario relativo ad ogni riga della matrice che definisce il carattere omino (vedi figura in basso).
Sarà necessario codificare in modo decimale la rappresentazione binaria delle righe e trasferire questi valori nelle rispettive locazioni di memoria tramite le istruzioni

```
80 FOR I=12448 TO 12455:READ A:POKE I,A:NEXT I
90 DATA 24,24,60,90,153,60,36,102
```

Il programma sarà concluso normalmente con l'istruzione END

```
100 END
```

Immettendo il comando RUN il programma sarà eseguito e al termine potremo constatare che il CBM-64 è in grado di riconoscere solo i 64 caratteri che abbiamo copiato nella RAM, e che sostituisce al carattere T la figura dell'omino.

colonna / **riga**	7	6	5	4	3	2	1	0	**binario**								**decimale**
0				X	X				0	0	0	1	1	0	0	0	24
1				X	X				0	0	0	1	1	0	0	0	24
2			X	X	X	X			0	0	1	1	1	1	0	0	60
3		X		X	X		X		0	1	0	1	1	0	1	0	90
4	X			X	X			X	1	0	0	1	1	0	0	1	153
5			X	X	X	X			0	0	1	1	1	1	0	0	60
6			X			X			0	0	1	0	0	1	0	0	36
7		X	X			X	X		0	1	1	0	0	1	1	0	102

Gestione degli sprites

L'uso dei soli caratteri definiti dall'utente non è in generale sufficiente per ottenere una buona grafica, soprattutto nel caso in cui si voglia far muovere oggetti sul video.
Infatti un oggetto, corrispondente ad uno o più caratteri definiti dall'utente, può muoversi sul video di un carattere alla volta e, per ottenere ciò è necessario che il programma sposti all'interno della memoria video tutti i caratteri costituenti l'oggetto.
Gli sprites sono insiemi di uno o più caratteri definiti dall'utente che possono essere spostati dal programma in modo semplice e tale da ottenere un movimento continuo sullo schermo.
Il CBM-64 è in grado di visualizzare contemporaneamente fino a 8 sprites (numerati da 0 a 7) che si muovono indipendentemente. Gli sprites hanno, a differenza dei caratteri, una matrice di definizione di 24 punti in larghezza (3 bytes) per 21 in altezza; quindi la loro definizione richiede uno spazio di memoria di 63 (21x3) bytes più un byte nullo alla fine per un totale di 64 bytes.
La matrice di definizione di uno sprite può essere memorizzata in una qualunque zona di memoria RAM e non necessariamente in posizione contigua alle matrici di definizione degli altri sprites, in quanto il CBM-64

ha un registro puntatore per ognuna di esse; i registri puntatori si trovano a partire dalla locazione 2040.
Incidentalmente, si noti che è possibile modificare semplicemente la forma di uno sprite cambiando il valore contenuto nel registro che punta alla matrice di definizione dello sprite, in modo tale da farlo puntare ad una definizione diversa dello stesso sprite. Così, se ad esempio si vuole rappresentare con uno sprite un omino che saluta alzando un braccio, si può pensare di memorizzare due diverse definizioni dello sprite, una a braccio alzato e l'altra a braccio abbassato e quindi puntare alternativamente alle due matrici: l'effetto sarà un'alternanza delle due forme dello stesso sprite, che così rappresenterà un omino che muove un braccio.
Nella figura sotto sono indicati i registri di controllo degli sprites con i relativi indirizzi di memoria.

ORDINAMENTO REGISTRI SPRITES

Indirizzo Basic VIC = 53248_{Dec} = $D000_{Esadec}$

Registri # Dec.	Esadec.	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
0	0	S0X7	S0X6	S0X5	S0X4	S0X3	S0X2	S0X1	S0X0	SPRITE 0 X
1	1	S0Y7							S0Y0	SPRITE 0 Y
2	2	S1X7							S1X0	SPRITE 1 X
3	3	S1Y7							S1Y0	SPRITE 1 Y
4	4	S2X7							S2X0	SPRITE 2 X
5	5	S2Y7							S2Y0	SPRITE 2 Y
6	6	S3X7							S3X0	SPRITE 3 X
7	7	S3Y7							S3Y0	SPRITE 3 Y
8	8	S4X7							S4X0	SPRITE 4 X
9	9	S4Y7							S4Y0	SPRITE 4 Y
10	A	S5X7							S5X0	SPRITE 5 X
11	B	S5Y7							S5Y0	SPRITE 5 Y
12	C	S6X7							S6X0	SPRITE 6 X
13	D	S6Y7							S6Y0	SPRITE 6 Y
14	E	S7X7							S7X0	SPRITE 7 X
15	F	S7Y7							S7Y0	SPRITE 7 Y
16	10	S7X8	S6X8	S5X8	S4X8	S3X8	S2X8	S1X8	S0X8	Bits alti del valore-X
17	11	RC8	EC5	BSM	BLNK	RSEL	YSCL2	YSCL1	YSCL0	
18	12	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	TAVOLA
19	13	LPX7							LPX0	LIGHT PEN X
20	14	LPY7							LPY0	LIGHT PEN Y
21	15	SE7							SE0	SPRITE ENABLE (IN/OUT)
22	16	N.C.	N.C.	RST	MCM	CSEL	XSCL2	XSCL1	XSCL0	
23	17	SEXY7							SEXY0	Espansione Y dello Sprite
24	18	VS13	VS12	VS11	CB13	CB12	CB11	CB10	N.C.	Memoriavideo
25	19	IRQ	N.C.	N.C.	N.C.	LPIRQ	ISSC	ISBC	RIRQ	Interrupt Request's
26	1A	N.C.	N.C.	N.C.	N.C.	MPLI	MISSC	MISBC	MRIRQ	Interrupt Request MASKS
27	1B	BSP7							BSP0	Priorità Sfondo Sprite
28	1C	SCM7							SCM0	Selezione Sprite Multicolore
29	1D	SEXX7							SEXX0	Espansione X dello Sprite
30	1E	SSC7							SSC0	COLLISIONE Sprite-Sprite
31	1F	SBC7							SBC0	COLLISIONE Sprite-Sfondo
		INFORMAZIONI-COLORE								
32	20									Margine
33	21									Sfondo 0
34	22									Sfondo 1
35	23									Sfondo 2
36	24									Sfondo 3
37	25	Sprites Multicolori								SMC0
38	26									SMC1
39	27									Colore Sprite 0
40	28									Colore Sprite 1
41	29									Colore Sprite 2
42	2A									Colore Sprite 3
43	2B									Colore Sprite 4
44	26									Colore Sprite 5
45	2D									Colore Sprite 6
46	2E									Colore Sprite 7

Una volta definito, uno sprite può essere visualizzato sullo schermo semplicemente assegnando il valore 0 o 1 ad un opportuno bit, relativo allo sprite in questione, nel registro di indirizzo 53269. Ad esempio, se si è definito lo sprite numero 2, esso può essere visualizzato ponendo a 1 il

terzo bit del registro 53269 con l'istruzione

POKE 53269, PEEK (53269) OR 4

e può essere cancellato ponendo a 0 lo stesso bit con l'istruzione

POKE 53269, PEEK (53269) AND 251

Uno sprite può essere visualizzato in un qualunque punto del video andando a porre in due registri associati allo sprite i valori delle posizioni X ed Y (ascissa ed ordinata) del punto: per visualizzare ad esempio lo sprite numero 2 nel punto 100,270 dello schermo, dato che i registri di posizionamento degli sprites sono posti a partire dall'indirizzo 53248, dovranno essere usate le seguenti istruzioni

POKE 53252,100 RETURN
POKE 53253,270 RETURN

Ricordiamo poi che i registri 53287 ÷ 53294 permettono di assegnare i colori desiderati agli sprites 0 ÷ 7 rispettivamente e che gli sprites possono essere estesi raddoppiandone altezza e/o larghezza. L'altezza può essere raddoppiata ponendo pari ad 1 il bit corrispondente allo sprite nel registro 53271 e la larghezza è estendibile operando allo stesso modo sul registro 53277. Il listato seguente ci fornisce un esempio di animazione di uno sprite formato da quattro linee verticali.

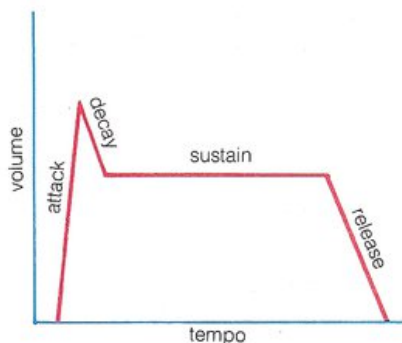
```
5 REM ANIMAZIONE DI UNO SPRITE
10 POKE 2040,13 : REM LEGGE DAL BLOCCO 13 I DATI
11 REM DELL'ANIMAZIONE
20 FOR I=0 TO 62
30 POKE 832+I,129
40 NEXT I : REM INSERISCE NEL BLOCCO 13 I DATI
41 REM DELL'ANIMAZIONE (13*63=832)
50 V=53248 : REM SI DISPONE ALL'INIZIO DEL CIRCUITO VIDEO
60 POKE V+21,1 : REM ABILITA L'ANIMAZIONE 0
70 POKE V+39,1 : REM IMPOSTA IL COLORE DELL'ANIMAZIONE 0
75 FOR I=100 TO 200 : REM ANIMAZIONE SPRITE
80 POKE V+1,I : REM IMPOSTA LA POSIZIONE Y DELL'ANIMAZIONE 0
90 POKE V+16,0 : POKE V,I : REM IMPOSTA LA POSIZIONE X
91 REM DELL'ANIMAZIONE 0
95 NEXT I
100 END
```

Il suono

Una delle caratteristiche più interessanti del CBM-64 è la sua capacità di generare suoni attraverso un programma che controlli gli appositi registri dedicati a questa funzione.
Il componente fondamentale che viene utilizzato per tali operazioni è il dispositivo di interfaccia del suono SID. Il SID contiene tre generatori di voci (cioè tre generatori di note) diversi, che combinati in modo opportuno consentono di ottenere dei suoni. Il SID contiene inoltre 25 locazioni di memoria (registri del suono), che controllano il volume, il tono ed il tipo di suono che viene emesso.
Nella tabella qui sotto sono riportate le locazioni di memoria dedicate a questo scopo, accanto alle rispettive funzioni.

Descrizione del registro del suono	Locazione di memoria		
	VOCE 1	VOCE 2	VOCE 3
Controllo frequenza voce (byte basso)	54272	54279	54286
Controllo frequenza voce (byte alto)	54273	54280	54287
Durata dell'impulso (byte basso)	54274	54281	54288
Durata dell'impulso (byte alto)	54275	54282	54289
Registro di controllo della forma d'onda	54276	54283	54290
Registro di controllo attack e decay	54277	54284	54291
Registro di controllo sustain e release	54278	54285	54292
		Funzioni di filtro	
Valore filtro cutoff (byte basso)		54293	
Valore filtro cutoff (byte alto)		54294	
Controllo risonanza filtro/ingresso voce		54295	
Controllo del volume		54296	

La locazione 54296 con i suoi 4 bits meno significativi controlla il volume del suono: si possono infatti ottenere 16 livelli di volume differenti che variano tra i valori 0 (mancanza di suono) e 15 (volume massimo). Per controllare il volume è sufficiente assegnare mediante una POKE uno di questi valori ai primi 4 bits della locazione 54296.
Per ottenere un suono è necessario assegnare adatti valori anche ai registri delle voci. Come si può notare, ognuna delle tre voci ha a disposizione sette locazioni di memoria, ciascuna delle quali controlla il tipo di suono prodotto da quella voce. Di queste locazioni di memoria, le prime due (54272-54273 per la prima voce) controllano la frequenza del suono emesso (compresa tra 1 e circa 3995 Hz) che viene codificata mediante parole di 16 bits. Nella figura della pagina seguente si possono vedere i valori da utilizzare per ottenere le diverse note.
La coppia successiva (54274-54275 per la prima voce) specifica la durata del suono emesso.
Le ultime tre locazioni di memoria (54276 ÷ 54278 per la prima voce) servono a determinare il timbro della nota specificando la forma dell'onda sonora. Infatti il primo dei tre registri permette di controllare l'attivazione o meno del generatore di suono, il tipo di onda sonora usata dal generatore (triangolare, identificata dal numero 17, quadra, dal numero 65, o a dente di sega, dal numero 33) oppure, non attivando il generatore di suono, l'emissione di un «rumore bianco», cioè un suono nel quale so-

REGISTRI DEL SUONO															
(Byte alto) Valore POKE	(Byte basso) Valore POKE	Frequenza (Hz)	Note musicali	(Byte alto) Valore POKE	(Byte basso) Valore POKE	Frequenza (Hz)	Note musicali	(Byte alto) Valore POKE	(Byte basso) Valore POKE	Frequenza (Hz)	Note musicali	(Byte alto) Valore POKE	(Byte basso) Valore POKE	Frequenza (Hz)	Note musicali
PRIMA OTTAVA				**TERZA OTTAVA**				**QUINTA OTTAVA**				**SETTIMA OTTAVA**			
1	12	16	DO	4	48	65	DO	16	195	262	DO	67	12	1046	DO
1	28	17	DO#	4	112	69	DO#	17	194	277	DO#	71	8	1109	DO#
1	45	18	RE	4	180	73	RE	18	208	294	RE	75	66	1175	RE
1	62	19	RE#	4	251	76	RE#	19	238	311	RE#	79	187	1244	RE#
1	81	21	MI	5	71	82	MI	21	30	330	MI	84	121	1319	MI
1	101	22	FA	5	151	87	FA	22	95	349	FA	89	127	1397	FA
1	123	23	FA#	5	237	92	FA#	23	180	370	FA#	94	209	1480	FA#
1	145	24	SOL	6	71	98	SOL	25	29	392	SOL	100	117	1568	SOL
1	169	25	SOL#	6	167	104	SOL#	26	155	415	SOL#	106	110	1661	SOL#
1	195	27	LA	7	12	110	LA	28	48	440	LA	112	194	1760	LA
1	221	29	LA#	7	119	117	LA#	29	221	466	LA#	119	118	1865	LA#
1	250	31	SI	7	233	123	SI	31	164	494	SI	126	145	1976	SI
SECONDA OTTAVA				**QUARTA OTTAVA**				**SESTA OTTAVA**				**OTTAVA OTTAVA**			
2	24	32	DO	8	97	131	DO	33	134	523	DO	134	24	2093	DO
2	56	34	DO#	8	225	139	DO#	35	132	554	DO#	142	17	2217	DO#
2	90	37	RE	9	104	147	RE	37	161	587	RE	150	132	2349	RE
2	125	39	RE#	9	247	156	RE#	39	221	622	RE#	159	119	2489	RE#
2	163	41	MI	10	143	165	MI	42	60	659	MI	168	242	2637	MI
2	203	44	FA	11	47	175	FA	44	191	698	FA	178	254	2794	FA
2	246	46	FA#	11	218	185	FA#	47	104	740	FA#	189	163	2960	FA#
3	35	49	SOL	12	142	196	SOL	50	58	784	SOL	200	234	3136	SOL
3	83	52	SOL#	13	77	208	SOL#	53	55	831	SOL#	212	220	3322	SOL#
3	134	55	LA	14	24	220	LA	56	97	880	LA	225	132	3520	LA
3	187	58	LA#	14	238	233	LA#	59	187	932	LA#	238	237	3729	LA#
3	244	62	SI	15	210	247	SI	63	72	988	SI	253	34	3951	SI

no presenti tutte le frequenze e che si presenta come un fruscio. Gli altri due registri specificano la forma dell'onda precisando la durata di quattro intervalli di tempo che la determinano (vedi grafico a lato). Per quanto riguarda la prima voce questi valori sono espressi sotto forma di 4 bits assegnati alla prima e alla seconda metà dei registri 54277 (attack, tempo di salita del suono al volume massimo, e decay, tempo di discesa del volume fino al valore medio) e 54278 (sustain, tempo di persistenza del suono a volume medio, e release, tempo di caduta a zero del volume). Infine i registri 54293 ÷ 54296 permettono di fissare una serie di parametri relativi al tipo dei suoni emessi dai tre generatori; questi parametri una volta fissati sono gli stessi per le tre voci e, tra gli altri, comprendono la regolazione del volume, di cui abbiamo parlato sopra.

Il Basic del CBM-64

Il CBM-64 è in grado di accettare e di eseguire programmi scritti in un «dialetto» del linguaggio Basic, il linguaggio più comunemente usato nella programmazione dei personal computer.
Come per un linguaggio comune, ad esempio l'italiano, esistono diverse varianti regionali (dialetti), così per un qualsiasi linguaggio di programmazione esistono, oltre ad una versione «ufficiale» detta **versione standard**, molte varianti particolari, generalmente una per ogni macchina.
I dialetti differiscono dalla versione standard per la completezza e per un certo numero di altre caratteristiche che, pur secondarie, sono comunque tali da non rendere eseguibili su una data macchina programmi scritti facendo riferimento al dialetto Basic di una macchina diversa.
La versione del Basic utilizzata dal CBM-64 è indicata col nome «COMMODORE 64 Basic V2», che appare sul video all'atto dell'accensione del computer.

L'interprete

Un programma è una successione ordinata di istruzioni che, sottoposte ad esecuzione, consentono di elaborare passo dopo passo un certo numero di dati, per arrivare alla soluzione di un determinato problema.
Nel Basic l'ordinamento della successione delle istruzioni è stabilito non dall'ordine fisico secondo cui esse vengono immesse, ma dalla numerazione che il programmatore impone.
Non è affatto necessario che i numeri di due istruzioni successive differiscano di una unità: le istruzioni potrebbero essere numerate, ad esempio, di dieci in dieci, e il computer riuscirebbe ugualmente ad eseguirle nella giusta successione, procedendo dal numero di linea più basso al più alto.
La pratica di numerare le istruzioni saltando ogni volta alcuni numeri risulta utile in quanto consente di inserire nuove istruzioni fra quelle già esistenti senza dover rinumerare tutto il programma. Dovendo inserire una nuova istruzione fra la 10 e la 20 sarà sufficiente digitare, **in qualsiasi punto del programma**, l'istruzione desiderata, dandole per esempio il numero 15. La nuova istruzione sarà collocata automaticamente al suo posto.
Bisogna chiarire subito che il microprocessore del CBM-64 non è in grado, da solo, di comprendere il significato delle frasi che il programmatore gli sottopone. Sappiamo infatti che i circuiti elettronici che compongono un computer sono in grado di elaborare solo segnali digitali binari, e non parole complesse. Il linguaggio Basic, come tutti i linguaggi di programmazione, è un espediente ideato per consentire ai programmatori di lavorare con parole più vicine alla sensibilità umana di quanto non lo siano gli aridi codici binari.
In realtà il microprocessore riceve le diverse istruzioni nella forma a lui direttamente comprensibile, cioè come successione di cifre 0 e 1. La traduzione delle parole utilizzate dal programmatore (**linguaggio simbolico**) nei corrispondenti codici binari (**linguaggio macchina**) è effettuata da un programma di sistema (cioè permanentemente contenuto nel CBM-64), chiamato **interprete**. Quando si manda in esecuzione un programma, l'interprete traduce ciascuna istruzione in linguaggio macchina, e la invia alla CPU che la esegue.
Ricordiamo dunque che le istruzioni da noi immesse nel CBM-64 saranno elaborate in primo luogo dall'interprete, e che è l'interprete a limitare l'uso delle parole-istruzione all'ambito previsto dal dialetto Basic utilizzato dalla macchina.
Le parole-chiave che l'interprete è in grado di riconoscere determinano l'esecuzione di diversi tipi di operazioni. A seconda del tipo di operazio-

ne attivato si parla di **comandi** e **istruzioni**.
I comandi impongono in genere al CBM-64 di operare in un certo modo **su un programma**, mentre le istruzioni operano **su dati e variabili**.

Dati, risultati, costanti e variabili

Un programma è una successione ordinata di istruzioni che agiscono su un certo numero di grandezze il cui valore è noto al programma (**dati**) per ricavare i valori di altre grandezze (**risultati**) che il programmatore vuole conoscere. Dati e risultati sono grandezze che il programma elabora.
Le grandezze che possono essere elaborate da un programma si suddividono in **costanti** e **variabili**: sono costanti le grandezze non soggette a variare durante l'esecuzione, mentre si dicono variabili le grandezze che possono assumere valori diversi nel corso di una stessa esecuzione.
Noi siamo abituati a pensare alle costanti e alle variabili come a grandezze numeriche, ma una delle caratteristiche più interessanti del Basic è quella di poter elaborare anche grandezze costituite da una successione di caratteri alfabetici, numerici e grafici. Queste grandezze sono chiamate «stringhe».
Una stringa può essere essa stessa una costante, se il suo valore non varia durante l'esecuzione del programma che la utilizza, o una variabile, se il programma la modifica in un modo qualsiasi.
Costanti e variabili (numeriche o di stringa) possono essere indicate all'interno di un programma utilizzando nomi simbolici. Ad esempio, se dobbiamo utilizzare i due numeri 10 e 20 per sommarli fra di loro e per visualizzare il risultato, potremmo immettere il programma riportato qui a fianco.

```
1    A=10
2    B=20
3    C=A+B
4    PRINT C
```

Le prime due istruzioni assegnano alla costante 10 il nome simbolico A e alla costante 20 il nome simbolico B. La terza calcola la somma (utilizzando i nomi simbolici) e la quarta la visualizza, utilizzando ancora un nome simbolico.
I nomi simbolici hanno una sintassi che dipende dal tipo di costante o di variabile cui si riferiscono.

```
A1=45.73
AB=12
CZ=129.992
```

- Per costanti e variabili numeriche **reali (in virgola mobile)** il nome simbolico deve essere composto al massimo da due caratteri, dei quali il primo necessariamente alfabetico. Alcuni esempi sono riportati qui a fianco. Variabili e costanti in virgola mobile possono assumere valori interi e decimali compresi nel campo $-10^{+38} \div +10^{+38}$, e occupano in memoria 4 bytes.

```
C1%=12
AC%=3
ZZ%=75
```

- Per costanti e variabili numeriche **intere** il nome deve essere composto come nel caso precedente, ma ad esso deve essere aggiunto in coda il simbolo %. Costanti e variabili intere possono assumere valori compresi nel campo $-32768 \div +32767$, e occupano in memoria 2 bytes.
  La minore occupazione di memoria ne rende conveniente l'uso, laddove possibile, in luogo delle variabili reali in virgola mobile.

```
A1$="MANUALI"
A2$="DI"
A3$="BASIC"
```

- Per costanti e variabili **di stringa** il nome deve essere composto come per costanti e variabili reali, ma ad esso deve essere aggiunto in coda il simbolo $ (dollaro). Costanti e variabili di stringa possono contenere al massimo 255 caratteri, ed occupano in memoria tanti bytes quanti sono i caratteri che le compongono (allocazione dinamica).

Il Basic del CBM-64 consente anche di definire un tipo particolare di variabili e di costanti, chiamato **array** (vettore, matrice). Un array è un insieme di variabili o costanti **omogenee** (cioè tutte dello stesso tipo) al quale si può fare riferimento con un nome simbolico unico.

Quando si devono elaborare tutti nello stesso modo dei dati omogenei risulta poco razionale utilizzare altrettanti nomi simbolici diversi.
Il Basic consente di evitare l'inconveniente facendo riferimento all'insieme dei dati come se si trattasse di un'unica variabile simbolica (array o variabile strutturata). Esiste un'istruzione (DIM) che consente di definire il nome simbolico di una variabile dichiarando al contempo di voler assegnare a quella variabile non uno, ma più valori distinti.
Ricevuta questa istruzione il calcolatore crea in memoria una sorta di tabella avente tante caselle quanti sono i dati che il programmatore intende assegnare, e numera queste caselle a partire da 0.
Dimensionato l'array (istruzione DIM) in ognuna delle caselle è possibile scrivere il valore di un dato, specificando il numero progressivo della casella nell'array (la numerazione delle caselle parte da 0).
Gli array possono avere anche più di una dimensione. Nel caso di array monodimensionale (**vettore**) la tabella creata in memoria si può immaginare come una semplice successione di caselle. Un array bidimensionale (**matrice**) si può riguardare come una tabella avente le caselle ripartite su più righe e colonne; per specificare una determinata casella serviranno due indici, il primo per la riga, il secondo per la colonna.
Un array a tre dimensioni può pensarsi costituito da più matrici impaginate l'una dopo l'altra in modo da costituire tante pagine di un libro. Per individuare una casella serviranno in questo caso tre indici, uno per indicare la riga, uno per la colonna e uno per la pagina.

Gli operatori

Gli operatori sono simboli che, inseriti nelle istruzioni e riconosciuti dall'interprete, attivano l'elaborazione delle variabili e delle costanti sulle quali agiscono. Gli operatori previsti dal Basic del CBM-64 sono di tre tipi.

- **Operatori aritmetici**. Sono i normali operatori dei calcoli aritmetici, e sono compendiati nella seguente tabella:

Operatore	Significato	Esempio
↑	elevamento a potenza	A↑2
/	divisione	A/2
*	moltiplicazione	A*2
+	somma	A+2
−	sottrazione	A−2

Se operatori aritmetici diversi sono impiegati nella stessa espressione il CBM-64 li attiva con la priorità con cui ogni simbolo compare nella tabella stessa (prima ↑, poi /, ecc.).
L'operatore aritmetico + può operare anche sulle stringhe, con il significato di concatenamento. Ad esempio, le linee indicate a fianco forniscono come risultato C$="CALCOLATORE"

```
10 LET A$="CALCO"
20 LET B$="LATORE"
30 LET C$=A$+B$
40 PRINT C$
```

- **Operatori relazionali**. Consentono di stabilire una comparazione fra due grandezze. Sono riassunti nella tabella che segue.

Operatore	Significato	Esempio
=	... uguale a...	A=B
>	... maggiore di...	A>B
<	... minore di...	A<B
>= (o =>)	... maggiore o uguale a...	A>=B
<= (o =<)	... minore o uguale a...	A<=B
<> (o ><)	... diverso da...	A<>B

- **Operatori logici**. Attivano l'esecuzione delle operazioni logiche su costanti e variabili. Qui sotto è riportata la tavola della verità degli operatori previsti dall'interprete del CBM-64.

A	B	A AND B	A OR B	NOT A	NOT B
0	0	0	0	1	1
0	1	0	1	1	0
1	0	0	1	0	1
1	1	1	1	0	0

Gli operatori logici sono utilizzati spesso all'interno delle istruzioni Basic, per controllare il verificarsi di particolari situazioni alle quali può essere condizionata l'esecuzione del programma.
Ad esempio, se si vuole condizionare un evento al fatto che risulti contemporaneamente A>B e A< >C si dovrà imporre la condizione

A>B AND A< >C

I comandi

Si è già detto che i comandi attivano funzioni che hanno per oggetto dei programmi. L'interprete del CBM-64 può riconoscere un numero limitato di comandi, che riguardano la scrittura, l'uso e il salvataggio dei programmi su supporto magnetico.
Prima di iniziare la scrittura di un nuovo programma è sempre opportuno ripulire la memoria di tutto ciò che contiene, che può essere un residuo indesiderato di precedenti elaborazioni (comando NEW). Durante la digitazione delle istruzioni è inoltre comodo poter visualizzare all'occorrenza le parti del programma che interessano (comando LIST). Terminata la digitazione si deve verificare il funzionamento del programma ordinandone l'esecuzione (comando RUN), dopo di che è possibile memorizzare il programma su supporto magnetico (SAVE), controllando che la copia sia fedele all'originale (VERIFY). Il programma può essere richiamato in memoria all'occorrenza utilizzando il comando LOAD.
C'è da notare che, per far risparmiare tempo nella digitazione di programmi e comandi, il CBM-64 offre all'utente la possibilità di abbreviare la maggior parte delle parole-chiave (se le abbreviazioni sono usate in una linea di programma, la parola-chiave apparirà nella forma completa). Le abbreviazioni sono indicate vicino o sotto le corrispondenti parole-chiave.
Vediamo ora in dettaglio il funzionamento dei comandi citati.

NEW

Cancella il contenuto della memoria accessibile all'utente. Qualsiasi informazione contenuta nella memoria è persa.

LIST
L SHIFT+I

Attiva la visualizzazione delle linee del programma presente in memoria al momento dell'immissione del comando.
Un comando LIST senza parametri visualizza l'intero programma, ma è possibile ordinare la visualizzazione delle parti che interessano specificando alcuni parametri. Ad esempio,

LIST 20 visualizza solo la linea 20
LIST 20 – 100 visualizza le linee da 20 a 100
LIST – 100 visualizza le linee dall'inizio alla 100
LIST 20 – visualizza le linee dalla 20 alla fine.

RUN
R SHIFT+U

Attiva l'esecuzione del programma che risiede in memoria all'atto dell'esecuzione del comando. È anche possibile ordinare l'esecuzione a partire da una certa linea in poi, specificando il numero della linea dopo la parola RUN.

CONT
C SHIFT+O

Riattiva l'esecuzione di un programma interrotta da un'istruzione STOP, da un'istruzione END o dalla pressione del tasto RUN/STOP.

SAVE
S SHIFT+A

Salva su supporto magnetico il programma contenuto nella memoria del CMB-64. L'operatore può assegnare al programma un nome (per un massimo di 16 caratteri), specificandolo tra doppi apici dopo la parola SAVE. Ad esempio, il comando SAVE "PROG.1" determina la memorizzazione su cassetta del programma residente in memoria; all'inizio della zona del nastro utilizzata il sistema scriverà il nome PROG.1, col quale il programma potrà essere richiamato (LOAD "PROG.1").
Specificando un ulteriore parametro (numero del dispositivo) è possibile effettuare la memorizzazione, anziché su cassetta, su floppy-disk. In questo caso il comando da inserire ha la sintassi SAVE "PROG.1", 8, dove il numero 8 indica il driver.

VERIFY
V SHIFT+E

Questo comando fa sì che il CMB-64 controlli il programma su nastro o su disco a fronte di quello presente in memoria. Ecco alcuni esempi:

VERIFY	controlla il primo programma su nastro
VERIFY "PROG.1"	cerca PROG.1 su nastro e lo confronta con quanto c'è in memoria
VERIFY "PROG.1", 8	cerca PROG.1 su disco (8) e lo controlla.

Il comando VERIFY è utile anche per posizionare la testina del registratore sul nastro dopo l'ultimo programma registrato, in modo da poter salvare il programma residente in memoria senza cancellarne altri. Basta immettere il comando VERIFY seguito dal nome dell'ultimo programma salvato: il CBM-64 dirà che non coincide con quello in memoria, ma alla fine la testina sarà posizionata sul nastro all'inizio della zona libera.

LOAD
L SHIFT+O

Attiva il caricamento in memoria di un programma salvato su nastro o su disco. Il funzionamento è duale rispetto a quello del comando SAVE.

LOAD	carica in memoria il primo programma che trova sul nastro
LOAD "PROG.1"	carica in memoria il programma PROG.1 residente su nastro
LOAD "PROG.1", 8	carica in memoria il programma PROG.1 residente su disco (8).

Le istruzioni

Le istruzioni che l'interprete Basic del CBM-64 è in grado di decodificare si possono suddividere in sei gruppi:

- istruzioni di assegnazione
- istruzioni di ingresso/uscita dati da consolle
- istruzioni di gestione dei files su supporto magnetico
- istruzioni di controllo
- istruzioni di definizione
- istruzioni di commento

Vediamo ora in dettaglio le istruzioni di ciascun gruppo.

■ Istruzioni di assegnazione

Si chiamano così perché tramite esse è possibile assegnare ad una o più variabili i valori che si desidera esse assumano.

LET
L SHIFT+E

Assegna un valore ad una variabile specificata col proprio nome simbolico; il valore può essere specificato esplicitamente o tramite un'espressione che può fare riferimento ai valori di altre variabili.

10 LET A=2	assegna alla variabile A il valore 2
10 LET A=B	assegna alla variabile A il valore della variabile B
10 LET A=B+2	assegna alla variabile A il valore ottenuto sommando 2 al valore della variabile B
LET A$="CIAO"	assegna il valore CIAO alla stringa A$.

Il codice LET è facoltativo: A=2 ha lo stesso significato di LET A=2.

CLR
C SHIFT+L

Assegna il valore 0 a tutte le variabili definite in precedenza. Le stringhe sono poste uguali alla stringa nulla (cioè alla stringa che non contiene alcun carattere). Se consideriamo il programma

```
10 LET A=7
20 LET B=3
30 CLR
```

dopo l'esecuzione della linea 30 avremo A=0, B=0.

POKE
P SHIFT+O

Permette di assegnare direttamente un valore numerico ad una cella di memoria. Il valore numerico da assegnare può anche essere specificato con un'espressione.

10 POKE 3844,3	assegna il valore 3 alla cella di indirizzo 3844
10 POKE 38444,A+B*4	assegna il valore A+B*4 alla cella di indirizzo 38444.

DATA
D SHIFT+A

Specifica un elenco di dati che il programma potrà leggere facendo uso dell'istruzione READ. L'impiego di questo metodo di assegnazione è molto utile quando un programma deve acquisire un certo numero di dati che rimangono gli stessi ad ogni esecuzione. Per quanto riguarda la sintassi, occorre specificare alcune cose sulle stringhe. La linea

```
10 DATA 3, 26, CIAO, 5, "COME VA?"
```

è predisposta per far leggere al programma i dati 3, 26, CIAO (stringa), 5, COME VA? (stringa). Come si può notare, la stringa CIAO non è racchiusa fra doppi apici, e questa è regola generale per qualunque stringa, a meno che essa non contenga i caratteri virgola o due punti.
In questi casi è necessario racchiudere la stringa fra doppi apici.

READ
R SHIFT+E

Permette di leggere sequenzialmente i dati specificati in un'istruzione DATA. Ad esempio, il programma

```
10 DATA 3, 26, CIAO, 5, "COME VA?"
20 READ A, B, C$, D, E$
```

assegna i seguenti valori: A=3, B=26, C$="CIAO", D=5, E$="COME VA?". Occorre prestare molta attenzione alla corrispondenza dei tipi.

RESTORE
RE SHIFT+S

Specifica che la lettura dei dati inclusi in un'istruzione DATA va effettuata nuovamente dall'inizio della lista. Ad esempio, il programma

```
10 DATA 3, 6, 36
20 READ A, B, C
30 RESTORE
40 READ D, E
```

attiva le seguenti assegnazioni: A=3, B=6, C=36, D=3, E=6.

■ Istruzioni di ingresso/uscita dati da consolle

Questo gruppo di istruzioni permette di impostare il colloquio tra il programma Basic nel quale sono inserite e l'operatore, rendendo possibile l'immissione di dati e la restituzione dei risultati dell'elaborazione.

INPUT

Causa l'attesa, da parte del programma, dell'immissione di dati da tastiera. L'istruzione è completata con uno o più nomi di variabili, e determina l'attesa di tanti dati quante sono le variabili citate. È anche possibile produrre la visualizzazione di un messaggio che chieda esplicitamente l'immissione o che ne specifichi le modalità. Vediamo alcuni esempi:

10 INPUT A	richiede l'immissione di un valore numerico da tastiera, da assegnare alla variabile A. All'atto dell'esecuzione della 10 sul video appare un punto interrogativo, e il programma rimane fermo alla linea 10 finché non viene immesso il dato richiesto
10 INPUT "VALORE DI A"; A	come nel caso precedente, ma stavolta è visualizzata la scritta VALORE DI A?
10 INPUT A, B, C$	attende tre dati da tastiera, due numerici e uno di stringa, da assegnare alle variabili A, B, C$

PRINT
?

Visualizza su video i valori assunti dalle variabili citate nella lista che segue la parola PRINT. Anche in questo caso è possibile visualizzare un messaggio esplicativo. Ad esempio,

10 PRINT A, B	visualizza i valori di A e B
10 PRINT "VALORE DI A=";A	visualizza la scritta VALORE DI A=seguita dal valore della variabile A
10 PRINT A, B, C, D$	visualizza i valori delle variabili nell'ordine con cui sono citate nella lista

GET
G SHIFT+E

Acquisisce un carattere da tastiera se è stato digitato. Il carattere immesso è assegnato ad una variabile di stringa specificata nell'istruzione. Ad esempio,

10 GET A$	se è presente un carattere digitato sulla tastiera, lo acquisisce e lo assegna alla variabile A$.

■ Istruzioni di gestione dei files su supporto magnetico

Consentono al microprocessore di accedere agli archivi (files) residenti su supporto magnetico (nastro, disco) e di utilizzarne il contenuto. Con questo insieme di istruzioni è possibile creare, modificare, leggere, scrivere i files. È importante notare che tutte le unità che compongono la configurazione base del CBM-64 (video, tastiera, registratore) sono viste dal processore come files (cioè come unità da cui possono essere lette, o verso le quali possono essere inviate le informazioni), ma per gestirle non è necessario utilizzare istruzioni particolari, dal momento che il CMB-64 è programmato per interpretare sempre comandi e istruzioni come se avessero per oggetto le suddette unità, a meno che non sia specificato diver-

samente dal programmatore.
Questi può deviare il flusso dei dati verso altri dispositivi, in particolare il driver e la stampante, utilizzando le istruzioni che seguono.

OPEN
O SHIFT+P

Consente di «aprire» il canale di comunicazione fra il computer e una periferica (stampante, driver...) o di rendere accessibile il contenuto di un file su supporto magnetico.
All'istruzione sono associati alcuni parametri che consentono al computer di individuare la periferica interessata.

- La parola OPEN è sempre seguita da un numero compreso fra 1 e 255, che è il numero col quale il programmatore vuole indicare la periferica. A questo numero dovranno fare riferimento eventuali altre istruzioni aventi per oggetto la comunicazione fra il programma e quella periferica. Ad esempio, se la OPEN apre un file su disco e gli assegna il numero X, ogni istruzione di lettura e di scrittura in quel file dovrà citare il numero X.
- Il primo numero è seguito da un secondo numero, separato dal primo con una virgola, che consente al CBM-64 di individuare il dispositivo oggetto della OPEN.
  La numerazione delle periferiche, stabilita dal costruttore, è la seguente:

Tastiera	0
Registratore	1
Interfaccia RS-232	2
Video	3
Stampante	4 o 5
Disk-driver	8

- Dopo il secondo numero può esserne inserito un terzo separato ancora con una virgola dal secondo, che ha significato diverso a seconda della periferica selezionata.
  L'impiego più comune riguarda il registratore e la stampante, nel qual caso il terzo numero ha i seguenti significati:

Registratore	0	lettura di un file da nastro
	1	scrittura in un file su nastro
	2	scrittura con indicatore di fine nastro al termine
Stampante	0	apre in modalità maiuscole/grafici
	7	apre in modalità testo.

- Dopo il terzo numero, separata da una virgola, può essere immessa una stringa, che assume significato diverso per le diverse unità:
  — se la OPEN ha per oggetto la stampante o il video la stringa viene inviata alla periferica come se si effettuasse una PRINT#
  — se la OPEN interessa il registratore la stringa è utilizzata come nome del file
  — se la OPEN è diretta al disco la stringa è utilizzata come nome del file o come codice di comando.

Per concludere la descrizione dell'istruzione OPEN consideriamo i seguenti esempi:

10 OPEN 9, 1	apre il canale verso il registratore (1) e indica questo dispositivo col numero 9
20 OPEN 12, 4	apre il canale verso la stampante (4) e indica questo dispositivo col numero 12
30 OPEN 1, 8, "FILE 1"	apre (o crea) un file di nome FILE 1 sull'unità disco (8) e indica questo file col numero 1.

CMD
C SHIFT+M

Dirotta l'esecuzione dei comandi e delle istruzioni che hanno per oggetto una periferica verso un dispositivo specificato. Questo dispositivo deve essere stato aperto in precedenza (OPEN) e il numero ad esso assegnato dal programmatore deve seguire la parola CMD. Ad esempio,

10 OPEN 1, 4	apre la stampante e le assegna il numero 1
20 CMD 1	dirotta sulla stampante comandi ed istruzioni che hanno per oggetto una periferica
30 PRINT A$	ogni istruzione PRINT è eseguita su carta anziché su video.

Nell'esempio, per ripristinare la normale esecuzione su video, è necessario chiudere il dispositivo 1 (stampante) con una CLOSE.

INPUT#
I SHIFT+N

Ha funzionamento analogo a quello della INPUT, ma accoglie i dati da un qualsiasi file (aperto con una OPEN) il cui numero sia citato dopo il codice #. Ad esempio,

10 OPEN 9, 1, 0, "FILE 1"	apre su nastro il file FILE 1 in lettura, e gli assegna il numero 9
20 INPUT#9, A$, B$	assegna alle variabili di stringa A$ e B$ il contenuto dei primi due records del file FILE 1

GET#

Analoga alla GET, consente di scrivere un byte in un file precedentemente aperto. La sintassi è spiegata dal seguente esempio:

10 OPEN 9, 1, 1, "FILE 1"	apre su nastro il file FILE 1 in scrittura e gli assegna il numero 9
20 LET A$="C"	
30 GET#9, A$	scrive sul primo byte libero di FILE 1 il carattere C

PRINT#
P SHIFT+R

Analoga alla PRINT, invia in scrittura una lista di dati (specificata nell'istruzione) su un dispositivo aperto in precedenza con una OPEN.
Al solito, il numero assegnato al dispositivo nella OPEN deve essere citato dopo il codice #. Ad esempio,

10 OPEN 9, 4	apre la stampante e le assegna il numero 9
20 LET A$="MANUALI "	
30 LET B$="DI "	
40 LET C$="BASIC"	
50 PRINT#9, A$; B$; C$	invia in stampa le tre stringhe; il risultato è MANUALI DI BASIC

CLOSE
CL SHIFT+O

Chiude un canale precedentemente aperto con una OPEN. Il numero assegnato al canale deve essere citato dopo la parola CLOSE. Ad es.,

10 OPEN 9, 4	apre la stampante e le assegna il numero 9
100 CLOSE 9	chiude la comunicazione con la stampante

■ Istruzioni di controllo

Sotto questa denominazione si raggruppano le istruzioni che, inserite in un programma, ne governano l'esecuzione, controllando il verificarsi di particolari condizioni e trasferendo di conseguenza il controllo alle diverse linee del programma.

FOR... TO... STEP...
NEXT...
F SHIFT+O...TO...
ST SHIFT+E...N SHIFT+E...

Sono le istruzioni che consentono lo svolgimento di un loop, cioè l'attivazione ricorsiva di una certa sequenza di istruzioni. La sequenza da ripetere deve essere inclusa fra le istruzioni FOR... TO... STEP... e NEXT..., specificando i limiti di variazione dell'indice del loop e il passo. Ad esempio, le linee

```
10 FOR I=2 TO 10 STEP 2 per I che va da 2 a 10 con passo 2...
20 PRINT I... sequenza da ripetere
30 NEXT I ripeti per il prossimo valore di I
```

determinano la stampa dei valori assunti dalla variabile I, e il risultato sarà la sequenza 2, 4, 6, 8, 10.
Se l'indice del loop deve variare con passo 1 non è necessario specificare il passo (STEP):

```
10 FOR I=1 TO 10
20 PRINT I
30 NEXT I
```

L'uscita sarà la sequenza 1, 2, 3, 4, ..., 10.
È utile ricordare che all'uscita di un loop il valore dell'indice risulta sempre incrementato dal passo rispetto all'ultimo valore utile.

IF... THEN...
IF... T SHIFT+H...

La parola IF deve essere seguita da una condizione, al verificarsi della quale deve essere eseguita l'istruzione che segue la parola THEN (SE... ALLORA...). Di seguito sono mostrati alcuni esempi:

```
10 IF A=B THEN C=0          se A=B allora poni C=0
10 IF A>B THEN GOTO 200     se A>B allora vai alla linea 200
10 IF A<B THEN PRINT "A MINORE DI B"
10 IF A>B AND A>0 THEN GOTO 120
```

GOTO...
ON... GOTO...
G SHIFT+O
ON... G SHIFT+O...

È la tipica istruzione di salto. Il controllo passa alla linea di programma specificata dopo la parola GOTO.
Nella prima forma il salto non è condizionato: se il programma incontra la linea 10 GOTO 1000 passa senz'altro ad eseguire la linea 1000.
La seconda forma consente invece di condizionare il salto al valore di una variabile.
La linea 10 ON A GOTO 10, 100, 1000 attiva il salto alla linea 10 se A=1, alla linea 100 se A=2, alla linea 1000 se A=3.

GOSUB...
GO SHIFT+S

È l'istruzione di chiamata ad una subroutine; la parola GOSUB deve essere seguita dal numero di linea alla quale la subroutine inizia.
Funziona in modo simile al GOTO, ma il sistema deve ricordare il punto dal quale è partito, perché, dopo aver eseguito la subroutine specificata, il controllo deve tornare alla linea che segue quella contenente la chiamata.

RETURN
RE SHIFT+T

È l'istruzione che conclude le subroutines e che determina la restituzione del controllo al programma chiamante.

STOP
S SHIFT+T

Determina l'arresto dell'esecuzione di un programma alla linea che contiene l'istruzione STOP, fino all'immissione da tastiera del comando CONT. Sul video appare la scritta BREAK ERROR IN LINE... seguita dal numero della linea contenente lo STOP.
Si impiega spesso per la correzione dei programmi, dal momento che, sotto interruzione, è possibile chiedere la stampa di tutte le variabili utilizzate dal programma.

END
E SHIFT+N

Conclude il programma, e serve per segnalare al sistema la fine dell'esecuzione. Il sistema risponde col messaggio READY.
L'istruzione END può essere utilizzata come lo STOP, con la differenza che sul video non appare alcun riferimento alla linea che la contiene.

■ Istruzioni di definizione

Sono le istruzioni che consentono al programmatore di definire funzioni e di dimensionare variabili strutturate (array).

DIM
D SHIFT+I

Permette di stabilire le dimensioni di un array. Ricordiamo che un array è un insieme di variabili omogenee cui si può fare riferimento con un nome simbolico unico e con uno o più indici.
Il dimensionamento di un array ad una dimensione (vettore) si ottiene con un'istruzione del tipo

10 DIM A$(10)	11 elementi di tipo stringa	(indice da 0 a 10)
10 DIM A(10)	11 elementi di tipo reale	(indice da 0 a 10)
10 DIM A% (10)	11 elementi di tipo intero	(indice da 0 a 10)

Il dimensionamento di un array a due dimensioni (matrice) con 3 righe e 5 colonne si ottiene nel modo

10 DIM A$(2,4)	15 elementi di tipo stringa	(indice di riga da 0 a 2, indice di colonna da 0 a 4)
10 DIM A(2,4)	15 elementi di tipo reale	(indice di riga da 0 a 2, indice di colonna da 0 a 4)
10 DIM A%(2,4)	15 elementi di tipo intero	(indice di riga da 0 a 2, indice di colonna da 0 a 4)

Il dimensionamento di un array a più dimensioni si ottiene in modo analogo: 10 DIM A$(2,4,3)

DEF FN
D SHIFT+E FN

Permette al programmatore di definire funzioni non previste nel linguaggio Basic. Ad esempio, la linea

10 DEF FNA (X)=X/2

definisce la funzione A(X)=X/2.
Il valore V che la funzione assume per un particolare valore di X, ad esempio 10, può essere chiamato nel programma con la semplice istruzione 20 V=FNA(10). Risulterà V=5.

■ Istruzioni di commento

Per migliorare la leggibilità dei programmi spesso risulta utile inserire tra le istruzioni alcuni commenti. Ciò può essere fatto in maniera molto semplice utilizzando l'istruzione REM.

Le funzioni

Le funzioni previste dal Basic del CBM-64 saranno qui suddivise, con riferimento al loro impiego nella programmazione più che alla struttura funzionale, secondo il seguente schema:

- funzioni numeriche
- funzioni matematiche
- funzioni di stringa
- funzioni di stampa
- funzioni di sistema

■ Funzioni numeriche

ABS

Restituisce il valore assoluto (valore privato del segno) di un'espressione: 10 A=ABS (−6/2) fornisce A=3.

INT

Restituisce il più grande intero minore o uguale al valore di un'espressione: 10 A=INT (−7.12) fornisce A=−8, mentre 20 A=INT (9.42) fornisce A=9.

SGN	Restituisce 1, 0, − 1 a seconda che il risultato di un'espressione sia positivo, nullo o negativo: 10 A=SGN (−9*2) fornisce A=−1.
ASC	Restituisce il codice ASCII (numerico) del primo carattere di una stringa. Ad esempio, se A$="TIPO" la linea 10 A=ASC(A$) fornisce A=84.
VAL	Restituisce il valore numerico di una stringa di cifre. Ad esempio, se A$="−125.3" (stringa), la 30 A=VAL (A$) fornisce A=−125.3 (valore numerico).
RND	Genera un numero casuale compreso fra 0 ed 1 (estremi esclusi): la linea 10 A=RND (0) fornisce in A un valore casuale compreso fra 0 e 1.

■ **Funzioni matematiche**

SIN	Calcola il seno trigonometrico dell'argomento, che deve essere espresso in radianti. Ad esempio, se X=3.14... (pi greco), la linea 10 A=SIN(X) fornisce A=0.
COS	Calcola il coseno trigonometrico dell'argomento, che deve essere espresso in radianti.
TAN	Calcola la tangente trigonometrica dell'argomento, che deve essere espressa in radianti.
ATN	Calcola l'arcotangente dell'argomento. Il risultato è espresso in radianti.
SQR	Calcola la radice quadrata dell'argomento, che non deve essere negativo.
EXP	Calcola l'esponenziale (e^x) di un numero x, che deve essere minore di 88.03, altrimenti si ha overflow.
LOG	Calcola il logaritmo naturale (o di Nepero, in base e) di un numero positivo.

■ **Funzioni di stringa**

CHR$	Restituisce il carattere corrispondente al codice ASCII specificato nell'argomento: 10 A$=CHR$ (84) fornisce A$="T".
STR$	Restituisce una stringa di caratteri uguali a quelli che compongono l'argomento. Ad esempio, se X=192.3 (numerico), 10 A$=STR$(X) fornisce A$="192.3" (stringa).
LEN	Calcola il numero di caratteri di una stringa. Ad esempio, se A$="BASIC", la linea 10 A=LEN(A$) fornisce A=5.
LEFT$	Estrae i caratteri più a sinistra di una stringa. Se A$="BASIC" la linea 10 B$=LEFT$(A$,2) estrae i due caratteri più a sinistra di A$, e fornisce B$="BA".
RIGHT$	Analoga alla precedente, estrae i caratteri più a destra, con la stessa sintassi.
MID$	Estrae i caratteri interni di una stringa: 10 B$=MID$(A$,2,3) estrae i caratteri centrali di A$ a partire dal secondo e per una lunghezza di 3 caratteri.

■ **Funzioni di stampa**

POS(X)	Restituisce la posizione corrente del cursore (colonna); X è un argomento fittizio, che può assumere un valore qualsiasi: 10 A=POS(X) restituisce in A il numero della colonna sulla quale è posizionato il cursore.

TAB

Posiziona il cursore su una determinata colonna dello schermo. Se il cursore si trova già a destra della colonna specificata il posizionamento è eseguito sulla riga successiva del video. È utilizzata sempre in una PRINT: la 10 PRINT TAB(10) posiziona il cursore in colonna 11 (la numerazione parte da 0).

SPC

Sposta il cursore verso destra. È utilizzata sempre in una PRINT: l'istruzione 10 PRINT SPC(10) sposta il cursore di 10 colonne.

■ **Funzioni di sistema**

PEEK

Restituisce il valore (decimale) contenuto in un byte di memoria specificato: 10 A=PEEK (36879) fornisce in A il contenuto della locazione di memoria di indirizzo 36879.

FRE(X)

Restituisce il numero di bytes di memoria liberi. L'argomento X è fittizio, e può assumere qualsiasi valore: 10 A=FRE(0) fornisce in A il numero di bytes liberi.

TI

Restituisce il tempo trascorso dall'accensione della macchina espresso numericamente in sessantesimi di secondo. La linea 10 PRINT TI/60 visualizza il numero di secondi trascorsi dall'accensione.

TI$

Restituisce una stringa di sei caratteri che contiene il tempo trascorso dall'accensione in ore, minuti, secondi: OOMMSS.

Avvertenza

A partire dal programma «La torre di Hanoi» (pag. 77) saranno utilizzati gli sprites, la bit-map e nuovi insiemi di caratteri grafici definiti dall'utente. Per inizializzare tutte le varie proprietà grafiche del CBM-64 è conveniente servirsi di quella parte di memoria in cui normalmente sono caricati i programmi utente.
Di conseguenza questi ultimi saranno caricati a partire dalla locazione di memoria di indirizzo 16384.
È quindi indispensabile, prima di digitare i listati presentati o di caricare i programmi dal supporto magnetico (disco o nastro), immettere i comandi

```
POKE 44,64        RETURN
POKE 16384,0      RETURN
```

Il primo (POKE 44,64) cambia il contenuto del registro 44 che punta all'inizio della zona dei programmi utente. Inserendo il valore 64 all'interno di questo registro, il sistema caricherà un qualsiasi programma che gli venga sottoposto a partire dalla locazione 16384, lasciando libere le locazioni di memoria di indirizzo più basso.
Il secondo comando (POKE 16384,0) azzera il contenuto della locazione iniziale della zona di memoria dove andrà caricato il programma.
Nel caso l'utente dimenticasse di inserire i due comandi POKE prima di iniziare a digitare o a caricare un programma, non riuscirà a farlo eseguire.
In tal caso non rimarrà che salvare nuovamente il programma su supporto magnetico (se non ce n'è già una copia), immettere i due comandi POKE e, per ultimo, ricaricare il programma, che finalmente potrà essere eseguito.
Ricordiamo che questa procedura è quella che si deve seguire sempre per qualsiasi programma che impieghi gli sprites, la grafica in modalità bit-map e nuovi insiemi di caratteri definiti dall'utente.

WHITBREAD
single drive
floppy disk

Poker d'assi

Ognuno di noi, almeno per sentito dire, conosce il gioco del poker. Vogliamo cercare di realizzare un programma che consenta ad un singolo giocatore di servirsi cinque carte scelte a caso da un ipotetico mazzo, di cambiarne alcune e di sapere dal CBM-64 quale punteggio ha ottenuto. Il nostro programma dovrà essere in grado di:

- presentare cinque carte coperte sul video
- scoprire le cinque carte
- aspettare che il giocatore comunichi quali carte vuole cambiare
- cambiare le carte scartate dal giocatore
- valutare il punteggio ottenuto.

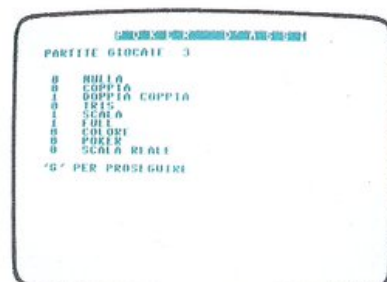
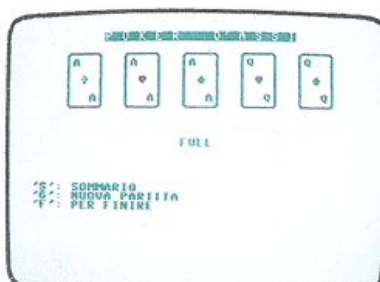
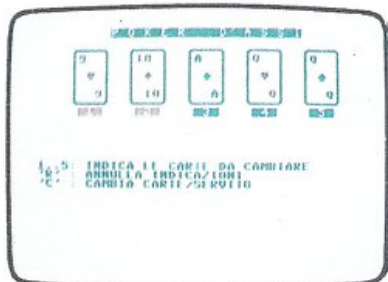
Se il giocatore lo desidera, il calcolatore presenterà il sommario dei punteggi ottenuti per poi riprendere l'esecuzione dall'inizio.

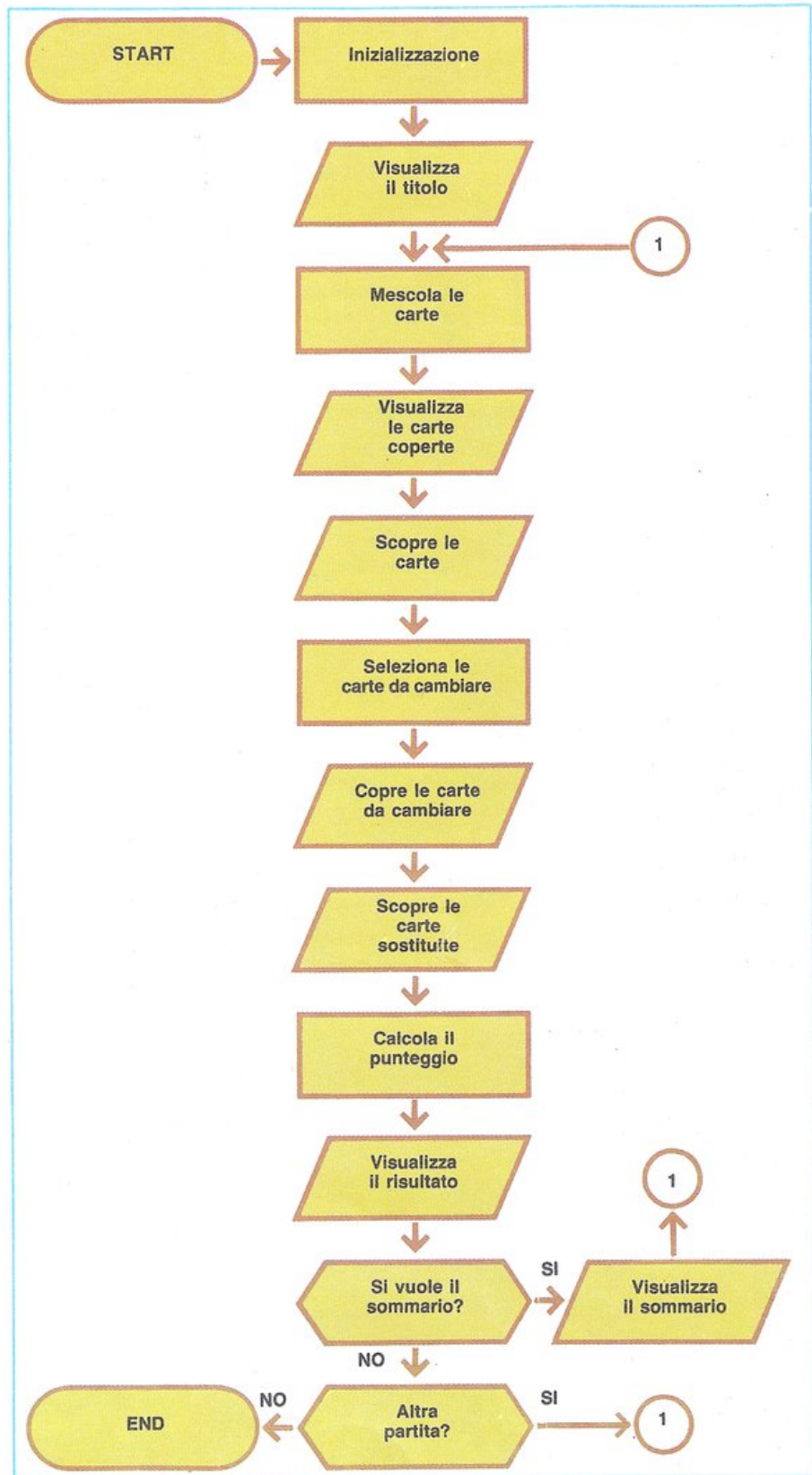
Analisi del problema

Il programma dovrà simulare il mescolamento di un mazzo di carte, dal quale saranno prelevate successivamente le carte da fornire inizialmente al giocatore e quelle utilizzate per cambiare le carte scartate. Il mazzo sarà rappresentato da due vettori VA e SE di 32 elementi ciascuno: i valori VA(I) e SE(I) rappresentano rispettivamente il valore e il seme dell'I-esima carta del mazzo.
Il mescolamento del mazzo potrà essere effettuato con scambi casuali degli elementi dei vettori. La presentazione delle carte distribuite al giocatore potrà avvenire

- disegnando sul video i contorni delle cinque carte
- colorando l'interno di grigio, per rappresentare il dorso delle carte
- sostituendo infine al grigio gli adatti simboli grafici per simulare lo scoprimento delle carte.

Il programma porrà poi i numeri 1, 2, 3, 4, 5 sul video in corrispondenza alle carte mostrate e il giocatore potrà specificare mediante il loro numero le carte che intende cambiare: queste carte saranno di nuovo coperte, sostituite e poi scoperte, visualizzando i nuovi valori che sostituiscono le carte cambiate.
Una volta calcolato e presentato sul video il punteggio ottenuto, il CBM-64 dovrà

- chiedere al giocatore se vuole il sommario dei punteggi ottenuti nelle partite precedenti
- in caso di risposta affermativa, visualizzare una tabellina che associa ad ogni punteggio del poker il numero di volte che tale punteggio è stato ottenuto nelle partite effettuate.

Diagrammi di flusso

Il mescolamento delle carte avviene con una sequenza di scambi di carte a coppie o, più esattamente, scambiando l'una dopo l'altra tutte le carte, a partire dalla prima, con altre carte scelte a caso. Il calcolo del punteggio si compie determinando dapprima se le carte sono disposte in scala, e quindi, in caso negativo, verificando la presenza degli altri punteggi, come la coppia, la doppia coppia, il tris, il full, il colore, il poker. Se si è verificata contemporaneamente la presenza di una scala a quella di un colore se ne deduce la presenza di una scala reale. Il risultato è presentato visualizzando il punteggio ottenuto, facendo lampeggiare il risultato sul video ed accompagnando la visualizzazione con un suono.

Il programma

Nella prima parte del programma è fissata la dimensione dei vettori VA e SE e degli altri vettori e matrici utilizzati nel corso del programma (linea 510). Le linee 560÷600 inizializzano i vettori VA e SE mentre le linee 640÷680 e 720÷740 inizializzano rispettivamente la matrice CC, contenente le coordinate degli spigoli delle carte, ed il vettore RI che contiene i risultati. La linea 810 presenta il titolo del programma. Le istruzioni 850÷940 controllano la visualizzazione dei cinque rettangoli che rappresentano le carte coperte (routine 2550 per i contorni delle carte e routine 2730 che colora di grigio l'interno della carta). L'istruzione 970 ci porta alla routine 2400 che simula il mescolamento delle carte. Le istruzioni 1010÷1140 visualizzano le cinque carte scoperte (routines 2550 e 2820). La linea 1130 serve a generare un breve ritardo per rendere il gioco più riflessivo (routine 2680). Le linee 1180÷1210 individuano ognuna delle 5 carte con numeri da 1 a 5 (routine 3510). Le istruzioni 1310÷1330 inizializzano il vettore FG che serve a contenere i numeri delle carte da cambiare;

```
100 REM ****************
110 REM **POKER D'ASSI**
120 REM ****************
140 REM VARIABILI UTILIZZATE
160 REM AP()        :MEMORIZZA LE PRIME CINQUE CARTE DEL MAZZO
170 REM FG()        :VETTORE CHE SPECIFICA LE CARTE DA CAMBIARE
180 REM BU()        :INDICA QUANTE COPPIE O TRIS COMPONGONO IL PUNTEGGIO OTTENUTO
190 REM SI          :CHIP GENERATORE DI NOTE
200 REM VA(),SE()   :CONTENGONO I VALORI ED I SEMI DEL MAZZO DI 32 CARTE
210 REM I,J,I1      :CONTATORI DI CICLO
220 REM CC()        :POSIZIONI SUL VIDEO DEGLI SPIGOLI DELLE CARTE
230 REM RI()        :MEMORIZZA LE OCCORRENZE DEI PUNTEGGI OTTENUTI
240 REM SF          :COLORE DEI CARATTERI DA VISUALIZZARE
250 REM A$          :STRINGA DI USO CORRENTE
260 REM PT          :POSIZIONE DELLA CARTA DA SOSTITUIRE A QUELLA CAMBIATA
270 REM BV,BS       :BUFFER TEMPORANEI
280 REM FC          :SE E' PARI A 1, LE CARTE DATE HANNO LO STESSO SEME
290 REM PU          :DA 0 A 8;PUNTEGGIO OTTENUTO
300 REM CS          :VARIABILE CONTENENTE UN NUMERO CASUALE
310 REM SE$         :SEME DELLA CARTA DA VISUALIZZARE
320 REM NU,NU$      :VALORE DELLA CARTA DA VISUALIZZARE
330 REM IM          :USATA NELLA ROUTINE DI ORDINAMENTO
340 REM FG          :SE E' PARI A 1 IL PUNTO PUO' ESSERE UNA SCALA
350 REM PF          :NUMERO DI PARTITE GIOCATE
351 REM VI          :CIRCUITO VIDEO DEL C64
352 REM PG$         :NOME PROGRAMMA
370 REM CARATTERI DI CONTROLLO
390 REM CHR$(147):CANCELLA IL VIDEO="]"
400 REM CHR$(144):CARATTERI DI COLORE NERO="■"
410 REM CHR$(19) :CURSORE IN ALTO A SINISTRA="■"
420 REM CHR$(18) :CARATTERI IN BIANCO SU NERO="■"
430 REM CHR$(17) :CURSORE IN BASSO DI UNA LINEA="■"
440 REM CHR$(29) :CURSORE A DESTRA DI UN CARATTERE="■"
450 REM CHR$(145):CURSORE IN ALTO DI UNA LINEA="]"
460 REM CHR$(157):CURSORE A SINISTRA DI UN CARATTERE="■"
470 REM CHR$(146):CARATTERI IN NERO SU BIANCO="■"
490 REM DIMENSIONO VARIABILI
510 DIM FG(4),AP(4),BU(1),VA(31),SE(31),CC(4,1),RI(8)
520 REM PRESENTAZIONE NOME PROGRAMMA
522 PG$="P O K E R  D' A S S I":GOSUB 62000
540 REM INIZIALIZZO I VETTORI VA(.) E SE(.) (VALORI E SEMI DELLE CARTE)
560 FOR I=0 TO 3
570 FOR J=0 TO 7
580 VA(I*8+J)=J+7
590 SE(I*8+J)=I
600 NEXT J:NEXT I
620 REM INIZIALIZZO MATRICE CC(.,.) (COORDINATE SPIGOLI DELLE CARTE)
640 CC(0,0)= 4:CC(0,1)=3
650 CC(1,0)=11:CC(1,1)=3
660 CC(2,0)=18:CC(2,1)=3
670 CC(3,0)=25:CC(3,1)=3
680 CC(4,0)=32:CC(4,1)=3
700 REM DIMENSIONO E INIZIALIZZO IL VETTORE DEI RISULTATI
720 FOR I=0 TO 8
730 RI(I)=0
740 NEXT I
760 REM INIZIA IL GIOCO
780 REM IMPOSTO IL COLORE DELLO SFONDO E DEI CARATTERI
800 POKE VI+32,1:POKE VI+33,1:PRINT CHR$(144);
810 PRINT CHR$(147);CHR$(18);TAB(9);"P O K E R   D' A S S I"
830 REM DISEGNO 5 CARTE BIANCHE UNA PER VOLTA CON LO SFONDO
850 FOR I=0 TO 4
870 REM CARTA BIANCA
890 GOSUB 2550
910 REM SFONDO CARTA
930 GOSUB 2730
940 NEXT I
955 REM MISCHIO LE CARTE
970 GOSUB 2400
990 REM DISEGNO, UNA PER VOLTA, LE PRIME CINQUE CARTE DEL MAZZO
1010 FOR I=0 TO 4
1030 REM CANCELLO LO SFONDO
1050 GOSUB 2550
1070 REM VISUALIZZO LA CARTA I-ESIMA
1090 GOSUB 2820
1110 REM RITARDO
1130 GOSUB 2680
1140 NEXT I
1160 REM ORA VISUALIZZO I NUMERI DA 1 A 5 SOTTO LE CINQUE CARTE
1180 SF=144
1190 FOR I=0 TO 4
1200 GOSUB 3510
1210 NEXT I
1230 REM VISUALIZZO SCRITTE
1245 PRINT CHR$(17);CHR$(17);CHR$(17);CHR$(17)
1250 PRINT "1..5: INDICA LE CARTE DA CAMBIARE"
1260 PRINT "'R' : ANNULLA INDICAZIONI"
1270 PRINT "'C' : CAMBIA CARTE/SERVITO"
1290 REM AZZERO VETTORE FG(.).I SUOI VALORI A 1 INDICANO CARTA DA CAMBIARE
1310 FOR I=0 TO 4
```

le istruzioni 1370÷1480 accettano un comando o il numero di una carta. Se il programma riceve il numero di una carta dal giocatore, lo stesso numero rappresentato sul video al di sotto della carta che si sta indicando è visualizzato in rosso per effetto delle istruzioni 1370÷1410. Le linee 1520÷1560 annullano le scritte sul video realizzate con le istruzioni PRINT (linee 1245÷1270). Le istruzioni 1600÷1690 cancellano dal video i numeri che individuano le carte da cambiare e pongono lo sfondo grigio sulle carte selezionate. Le linee 1730÷1800 realizzano il cambio delle carte selezionate con le prime carte del mazzo prese dai vettori VA ed SE e le 1840÷1860 visualizzano le nuove carte prese dal mazzo.
Le istruzioni 1930÷1960 determinano se si è ottenuto come punteggio il colore.
Le linee 2030÷2050 memorizzano i valori delle carte visualizzate e la 2090 chiama la routine 3710 che determina il punteggio ottenuto.
Le 2130, 2140 controllano se il punto realizzato è una scala reale, l'istruzione 2220 porta alla routine 4240 che visualizza il punteggio ottenuto. Le istruzioni 2255÷2330 chiedono al giocatore se vuole un sommario dei punteggi ottenuti nelle partite giocate: se la risposta è sì lo visualizzano chiamando il sottoprogramma alla linea 4600, altrimenti l'utente può terminare o giocare di nuovo.
In coda al programma si trovano due routines (linea 61000 e linea 62000): la

```
1320 FG(I)=0
1330 NEXT I
1350 REM NUMERO SOTTO LA CARTA I-ESIMA IN NERO O ROSSO A SECONDA DEL VALORE FG(I
)
1370 FOR I=0 TO 4
1380 SF=144
1390 IF FG(I)=1 THEN SF=150
1400 GOSUB 3510
1410 NEXT I
1430 REM ORA LEGGO IL COMANDO
1450 GET A$:IF A$="" THEN GOTO 1450
1460 IF A$>="1" AND A$<="5" THEN FG(VAL(A$)-1)=1:GOTO 1370
1470 IF A$="R" THEN GOTO 1310
1480 IF A$<>"C"THEN GOTO 1450
1500 REM ORA CANCELLO LE TRE RIGHE CHE RICHIEDEVANO UN COMANDO
1520 PRINT CHR$(19);
1530 FOR I=1 TO 15:PRINT:NEXT I
1540 FOR I=1 TO 3
1545 REM STAMPO 35 SPAZI
1550 PRINT "                                   "
1560 NEXT I
1580 REM ORA CANCELLO I NUMERI SOTTO LE CARTE. LI DISEGNO CON IL COLORE BIANCO
1600 SF=5
1610 FOR I=0 TO 4
1620 GOSUB 3510
1630 NEXT I
1650 REM ORA DISEGNO LO SFONDO DELLE CARTE INDICATE CON 1 IN FG(.)
1670 FOR I=0 TO 4
1680 IF FG(I)=1 THEN GOSUB 2730
1690 NEXT I
1710 REM ORA CAMBIO LE CARTE. PRIMA NEI VETTORI VA(.) E SE(.), E POI SUL VIDEO
1730 PT=5
1740 FOR I=0 TO 4
1750 IF FG(I)=0 THEN GOTO 1800
1760 BV=VA(I):BS=SE(I)
1770 VA(I)=VA(PT):SE(I)=SE(PT)
1780 VA(PT)=BV:SE(PT)=BS
1790 PT=PT+1
1800 NEXT I
1820 REM ORA VISUALIZZO LE CARTE CAMBIATE SUL VIDEO
1840 FOR I=0 TO 4
1850 IF FG(I)=1 THEN GOSUB 2680:GOSUB 2550:GOSUB 2820
1860 NEXT I
1880 REM ORA CALCOLO IL PUNTEGGIO OTTENUTO
1910 REM CONTROLLO SE LE CARTE SONO TUTTE DELLO STESSO COLORE (FC=1)
1930 FC=1
1940 FOR I=1 TO 4
1950 IF SE(I)<>SE(0) THEN FC=0
1960 NEXT I
1980 REM SE FC=1 LE CARTE SONO TUTTE DELLO STESSO COLORE
2010 REM ORA COPIO I VALORI DELLE CARTE VISUALIZZATE NEL VETTORE AP(.)
2030 FOR I=0 TO 4
2040 AP(I)=VA(I)
2050 NEXT I
2070 REM CONTROLLO IL PUNTEGGIO OTTENUTO
2090 GOSUB 3710
2110 REM CONTROLLO SE HO FATTO SCALA REALE O COLORE
2130 IF FC=1 AND PU=4 THEN PU=8
2140 IF FC=1 AND PU=0 THEN PU=6
2160 REM MEMORIZZO PUNTEGGIO OTTENUTO
2180 RI(PU)=RI(PU)+1
2200 REM STAMPO IL RISULTATO
2220 GOSUB 4240
2240 REM ORA CHIEDO SE SI VUOLE IL SOMMARIO OD ALTRO
2255 PRINT:PRINT:PRINT:PRINT:PRINT
2260 PRINT "'S': SOMMARIO          "
2270 PRINT "'G': NUOVA PARTITA"
2280 PRINT "'F': PER FINIRE";
2290 GET A$:IF A$="" THEN GOTO 2290
2300 IF A$="G" THEN GOTO 810
2310 IF A$="S" THEN GOSUB 4600:GOTO 810
2320 IF A$="F" THEN POKE 53280,14:POKE 53281,6:PRINT "[illegible]":END
2330 GOTO 2290
2350 REM INIZIANO LE SUBROUTINES
2380 REM SUBROUTINE CHE MESCOLA DUE VOLTE LE CARTE NEI VETTORI VA(.) ED SE(.).
2400 CS=RND(-TI)*1000
2410 FOR J=0 TO 1
2420 FOR I=0 TO 31
2430 BV=VA(ABS(31*J-I)):BS=SE(ABS(31*J-I))
2440 CS=RND(CS)*32
2450 SE(ABS(31*J-I))=SE(CS)
2460 VA(ABS(31*J-I))=VA(CS)
2470 VA(CS)=BV
2480 SE(CS)=BS
2490 NEXT I
2500 NEXT J
2510 RETURN
2530 REM SUBROUTINE: DISEGNO UNA CARTA BIANCA NELLA LOCAZIONE I-ESIMA
2550 PRINT CHR$(19);
2560 FOR J=1 TO CC(I,1)-1:PRINT:NEXT J
```

prima serve ad inizializzare le costanti e la seconda stampa il titolo del programma.

```
2570 PRINT TAB(CC(I,0));"╭────╮"
2580 PRINT TAB(CC(I,0));"|    |"
2590 PRINT TAB(CC(I,0));"|    |"
2600 PRINT TAB(CC(I,0));"|    |"
2610 PRINT TAB(CC(I,0));"|    |"
2620 PRINT TAB(CC(I,0));"|    |"
2630 PRINT TAB(CC(I,0));"╰────╯"
2640 RETURN
2660 REM SUBROUTINE: RITARDO
2680 FOR J=1 TO 300:NEXT J
2690 RETURN
2710 REM SUBROUTINE: DISEGNO DORSO DELLA CARTA NELLA LOCAZIONE I-ESIMA
2730 PRINT CHR$(19);
2740 FOR J=1 TO CC(I,1):PRINT:NEXT J
2750 FOR J=1 TO 5
2760 PRINT TAB(CC(I,0)+1);"▒▒▒▒"
2770 NEXT J
2780 RETURN
2800 REM SUBROUTINE: DISEGNO LA CARTA CA-ESIMA NELLA POSIZIONE I-ESIMA
2820 PRINT CHR$(19);
2830 FOR J=1 TO CC(I,1)-1:PRINT:NEXT J
2840 PRINT TAB(CC(I,0));
2860 REM CONTROLLO IL SEME DELLA CARTA
2880 IF SE(I)=0 THEN SE$="▒♥"
2890 IF SE(I)=1 THEN SE$="▒♦"
2900 IF SE(I)=2 THEN SE$="█♣"
2910 IF SE(I)=3 THEN SE$="█♠"
2930 REM DISEGNO IL SEME DELLA CARTA
2950 PRINT CHR$(17);CHR$(17);CHR$(17);CHR$(29);CHR$(29);
2960 PRINT SE$;
2980 REM DISEGNO IL NUMERO DELLA CARTA
3010 REM MI POSIZIONO SULLO SPIGOLO (IN ALTO A SINISTRA) DELLA CARTA
3030 PRINT CHR$(145);CHR$(145);CHR$(157);CHR$(157);
3050 REM CALCOLO E STAMPO IL NUMERO (VALORE) DELLA CARTA
3070 NU=VA(I)
3080 IF NU<=10 THEN NU$=STR$(NU)
3090 IF NU=11 THEN NU$=" J"
3100 IF NU=12 THEN NU$=" Q"
3110 IF NU=13 THEN NU$=" K"
3120 IF NU=14 THEN NU$=" A"
3140 REM TOLGO LO SPAZIO BIANCO IN TESTA A NU$
3160 NU$=RIGHT$(NU$,LEN(NU$)-1)
3170 PRINT NU$;
3190 REM ORA LO STAMPO NELLO SPIGOLO IN BASSO A DESTRA
3220 REM SE IL NUMERO E' 10,INDIETRO DI UNO SPAZIO
3240 IF NU$="10" THEN PRINT CHR$(157);
3250 PRINT CHR$(17);CHR$(17);CHR$(17);CHR$(17);CHR$(29);
3270 REM SE ORA IL NUMERO E' 10, DEVO SPOSTARMI DI UNO SPAZIO A SINISTRA
3290 IF NU$="10" THEN PRINT CHR$(157);
3300 PRINT NU$;
3320 REM ORA REIMPOSTO I CARATTERI NERI
3340 PRINT CHR$(144);
3360 REM ORA EMETTO UN SUONO DI FREQUENZA PROPORZIONALE AL VALORE DELLA CARTA
3380 POKE SI+24,15
3390 POKE SI,0
3400 POKE SI+5,0
3410 POKE SI+6,240
3420 POKE SI+1,10*(NU-6)
3430 POKE SI+4,0:POKE SI+4,17
3440 FOR I1=1 TO 200:NEXT I1
3450 POKE SI+4,0
3460 POKE SI+24,0
3470 RETURN
3490 REM SUBROUTINE: VISUALIZZO IL NUMERO I+1 SOTTO LA CARTA I-ESIMA
3510 PRINT CHR$(19);TAB(CC(I,0));
3530 REM IL CURSORE E' SULL'ANGOLO IN ALTO A SINISTRA DELLA CARTA I-ESIMA
3550 PRINT CHR$(SF);
3560 FOR I1=1 TO 9
3570 PRINT CHR$(17);
3580 NEXT I1
3590 PRINT " ";CHR$(18);STR$(I+1);" ";CHR$(146)
3610 REM RIPRISTINO IL COLORE NERO PER I CARATTERI
3630 PRINT CHR$(144);
3640 RETURN
3660 REM SUBROUTINE: DETERMINO PUNTEGGIO
3690 REM ORDINO IL VETTORE AP(.)
3710 FOR I=0 TO 3
3720 IM=I
3730 FOR J=I TO 4
3740 IF AP(J)<AP(IM) THEN IM=J
3750 NEXT J
3770 REM SCAMBIO IL VALORE IN POSIZIONE IM CON QUELLO IN POSIZIONE I
3790 BV=AP(I)
3800 AP(I)=AP(IM)
3810 AP(IM)=BV
3820 NEXT I
3840 REM CONTROLLO SE E' SCALA
3860 FG=1
3870 FOR I=0 TO 2
3880 IF AP(I+1)<>AP(I)+1 THEN FG=0
```

```
3890 NEXT I
3900 IF FG=0 THEN GOTO 3960
3920 REM I PRIMI QUATTRO VALORI SONO IN SCALA. CONTROLLO IL QUINTO
3940 IF (AP(4)=AP(3)+1) OR (AP(4)=14 AND AP(0)=7) THEN PU=4:RETURN
3960 REM CONTO SE CI SONO RIPETIZIONI DELLO STESSO VALORE
3970 REM COPPIA,DOPPIA COPPIA,TRIS,FULL,POKER
4000 REM DATO CHE IL VETTORE E' ORDINATO,VALORI UGUALI SONO CONSECUTIVI
4020 BU(0)=0:BU(1)=0
4030 J=0
4040 FOR I=0 TO 3
4050 IF AP(I+1)=AP(I) THEN BU(J)=BU(J)+1
4060 IF AP(I+1)<>AP(I) AND BU(0)<>0 THEN J=1
4070 NEXT I
4090 REM ORA IN BU(0) E BU(1) HO IL PUNTEGGIO RAGGIUNTO LO CALCOLO
4110 IF BU(0)=0 AND BU(1)=0 THEN PU=0:RETURN
4120 IF BU(0)=1 AND BU(1)=0 THEN PU=1:RETURN
4130 IF BU(0)=1 AND BU(1)=1 THEN PU=2:RETURN
4140 IF BU(0)=2 AND BU(1)=0 THEN PU=3:RETURN
4150 IF BU(0)=2 AND BU(1)=1 THEN PU=5:RETURN
4160 IF BU(0)=1 AND BU(1)=2 THEN PU=5:RETURN
4170 IF BU(0)=3 AND BU(1)=0 THEN PU=7:RETURN
4190 REM UNO DEGLI IF PRECEDENTI E' SICURAMENTE VERIFICATO
4220 REM SUBROUTINE: VISUALIZZAZIONE DEL RISULTATO
4240 PRINT CHR$(19);
4250 FOR I=1 TO 12:PRINT:NEXT I
4260 IF PU=0 THEN RETURN
4270 IF PU=1 THEN A$="COPPIA"
4280 IF PU=2 THEN A$="DOPPIA COPPIA"
4290 IF PU=3 THEN A$="TRIS"
4300 IF PU=4 THEN A$="SCALA"
4310 IF PU=5 THEN A$="FULL"
4320 IF PU=6 THEN A$="COLORE"
4330 IF PU=7 THEN A$="POKER"
4340 IF PU=8 THEN A$="SCALA REALE"
4360 REM FACCIO LAMPEGGIARE IL RISULTATO ED EMETTO UN SUONO
4380 POKE SI+24,15
4390 POKE SI,0
4400 POKE SI+5,0
4410 POKE SI+6,240
4420 FOR I=1 TO 20
4430 PRINT TAB((40-LEN(A$))/2);CHR$(18);A$
4440 PRINT CHR$(145);
4450 POKE SI+1,40
4460 POKE SI+4,0:POKESI+4,17
4470 FOR J=1 TO 50:NEXT J
4480 PRINT TAB((40-LEN(A$))/2);A$
4490 PRINT CHR$(145);
4500 POKE SI+1,140
4510 POKE SI+4,0
4520 POKE SI+4,17
4530 FOR J=1 TO 100:NEXT J
4540 POKE SI+4,0
4550 NEXT I
4555 POKE SI+24,0
4560 RETURN
4580 REM SUBROUTINE: VISUALIZZO SOMMARIO
4600 PF=0
4620 REM CONTO PARTITE FATTE
4640 FOR I=0 TO 8
4650 PF=RI(I)+PF
4660 NEXT I
4670 PRINT CHR$(147);
4680 PRINT TAB(9);CHR$(18);"P O K E R   D' A S S I"
4690 PRINT:PRINT "PARTITE GIOCATE:";PF
4700 PRINT:PRINT
4710 PRINT RI(0);"  NULLA"
4720 PRINT RI(1);"  COPPIA"
4730 PRINT RI(2);"  DOPPIA COPPIA"
4740 PRINT RI(3);"  TRIS"
4750 PRINT RI(4);"  SCALA"
4760 PRINT RI(5);"  FULL"
4770 PRINT RI(6);"  COLORE"
4780 PRINT RI(7);"  POKER"
4790 PRINT RI(8);"  SCALA REALE"
4800 PRINT:PRINT "'G' PER PROSEGUIRE";
4810 GET A$:IF A$<>"G" THEN GOTO 4810
4820 RETURN
60960 REM SUBROUTINE:INIZIALIZZAZIONE COSTANTI
60980 REM CIRCUITO VIDEO
61000 VI=53248
61020 REM CIRCUITO SUONO
61040 SI=54272
61060 REM MEMORIA VIDEO
61080 MV=1024
61100 REM MEMORIA COLORE
61120 MC=55296
61180 REM INIZIALIZZAZIONE CHIP SUONO
61200 FOR I=0 TO 24
61210 POKE SI+I,0
61220 NEXT I
```

```
61230 RETURN
61980 REM SUBROUTINE:STAMPA IL TITOLO DEL PROGRAMMA MEMORIZZATO IN PG$
62000 GOSUB 61000
62010 POKE VI+32,15
62020 POKE VI+33,15
62030 PRINT "[illegible]";
62040 PRINT TAB(6);"┌──────────────────────────┐"
62050 FOR I=1 TO 5
62060 PRINT TAB(6);"│                          │"
62070 NEXT I
62080 PRINT TAB(6);"└──────────────────────────┘"
62090 PRINT TAB(6);"[illegible]DIGITA [illegible]RETURN[illegible] PER PROSEGUIRE"
62100 PRINT "[illegible]"
62110 FOR I=1 TO 5
62120 PRINT TAB(7);
62130 FOR J=1 TO 26
62140 PRINT "[illegible] ";
62150 NEXT J
62160 PRINT
62170 NEXT I
62190 REM ORA SCRIVO IL TITOLO
62210 PRINT "[illegible]":TAB((40-LEN(PG$))/2);"[illegible]";PG$
62220 GET Z9$
62230 IF Z9$<>CHR$(13) THEN GOTO 62220
62240 PRINT "[illegible]";
62250 RETURN
```

Un medico poco serio

Questo progetto porterà alla realizzazione di un gioco che simulerà il comportamento di un azzeccagarbugli della medicina. Il programma si comporterà come un medico (non molto affidabile) dal quale un paziente si reca per un consulto. Esso porrà al paziente alcune domande per individuare la fantasiosa sindrome che lo affligge; alla fine presenterà una diagnosi e suggerirà una cura, insieme alle controindicazioni per la cura proposta. La finalità è quella di giocare con il computer e quindi non è il caso di prendere sul serio la diagnosi fornita.

Analisi del problema

All'inizio, il programma si dovrà presentare e dovrà chiedere le classiche informazioni necessarie per predisporre una cartella clinica: cognome, nome, età e sesso del paziente e tipo di disturbi accusati. Quindi dovrà sottoporre al paziente alcune domande, scelte in funzione dei disturbi che egli accusa; sulla base dell'anamnesi che ne consegue potrà finalmente costruire il responso, corredandolo della cura che ritiene più opportuna, nonché evidenziare le controindicazioni esistenti per la cura proposta. Terminerà, da buon luminare della medicina, presentando la parcella, che sarà calcolata sulla base del tipo di disturbi presentati dal paziente.
Il programma dovrà elaborare tre tipi di frasi: le domande iniziali, le frasi che serviranno per costruire il responso e le frasi introduttive che precederanno ogni sezione del responso. Poiché le frasi di ciascun tipo sono omogenee tra loro, utilizzeremo per contenerle tre matrici, in particolare DOM$ per le domande, MAL$ per le frasi introduttive e DIA$ per le frasi che costituiranno le varie parti del responso.
Le domande iniziali saranno tali da individuare solo sei tipi di disturbi possibili: sessuali (maschili e femminili), respiratori, gastrici, nervosi e cardiaci. È chiaro che le frasi da utilizzare saranno diverse per ciascun tipo di disturbo, per cui dovremo dividere sia le domande che le frasi per la diagnosi in sei gruppi, uno per ogni tipo di disturbo.
Iniziamo dunque con l'esame degli array preposti al dialogo interattivo fra medico e paziente.
Poiché le domande risultano logicamente divise in gruppi conviene utilizzare come struttura dati una matrice anziché un vettore. Se definiamo dieci domande per ogni disturbo possiamo usare la matrice DOM$ (6, 10), in cui il primo indice, che assumerà i valori tra 1 e 6, corrisponderà al tipo di disturbo, mentre il secondo indice, che varierà da 1 a 10, individuerà ogni singola domanda (vedi grafico in basso).
Dopo l'immissione del tipo di disturbo il programma presenterà al paziente tre domande scelte fra le dieci relative al disturbo denunciato.

DOMANDE POSSIBILI
Ogni riga contiene le domande relative ad un determinato tipo di disturbo

DOMANDE POSTE

RISPOSTE FORNITE

Gli indici di colonna delle domande poste al paziente sono memorizzati, nell'ordine, in PD. I codici delle risposte fornite dal paziente (1 = SI, 2 = NO, 3 = POCO/TALVOLTA) sono memorizzati in Z nell'ordine secondo cui sono poste le domande. Ad esempio, Z(1) conterrà la risposta fornita dal paziente alla domanda il cui indice è in PD(1).

A questo punto si rende necessario gestire le risposte fornite dal paziente utilizzando due semplici vettori (visibili a fianco) la cui utilità risulterà chiara allorché si parlerà della logica di selezione del responso.
Il responso che il CBM-64 costruirà alla fine dell'anamnesi sarà composto di tre parti: la definizione della malattia riscontrata, la cura da seguire e le controindicazioni alla cura precedentemente consigliata. Faremo precedere ciascuna delle tre parti da una frase introduttiva.
Per memorizzare le frasi che utilizzeremo per costruire il responso useremo dunque una matrice più complessa, DIA$(6, 10, 3), in cui primo e secondo indice sono analoghi a quelli della matrice DOM$, mentre il terzo, con valori da 1 a 3, individuerà il tipo delle frasi: per la diagnosi, per la cura e per le controindicazioni. Infine memorizzeremo le frasi di introduzione alle tre parti che costituiscono il responso nella matrice MAL$(3, 4), in cui il primo indice, con valori da 1 a 3, individuerà il blocco cui si riferiscono le frasi introduttive (diagnosi, cura, controindicazioni), mentre il secondo indice distinguerà ogni singola frase.

FRASI INTRODUTTIVE

RESPONSO

La frase che si trova, ad esempio, in posizione 4,5 descrive una malattia che dovrebbe corrispondere al sintomo contenuto nella corrispondente posizione di DOM$. Ovviamente la corrispondenza è solo ironica.

La visualizzazione del responso sarà organizzata così:

Diagnosi

1 / Il programma sceglie casualmente una frase introduttiva dalla prima riga di MAL$
2 / Il programma sceglie una malattia dalla prima pagina di DIA$ in dipendenza della risposta fornita dal paziente
3 / Stampa delle frasi selezionate

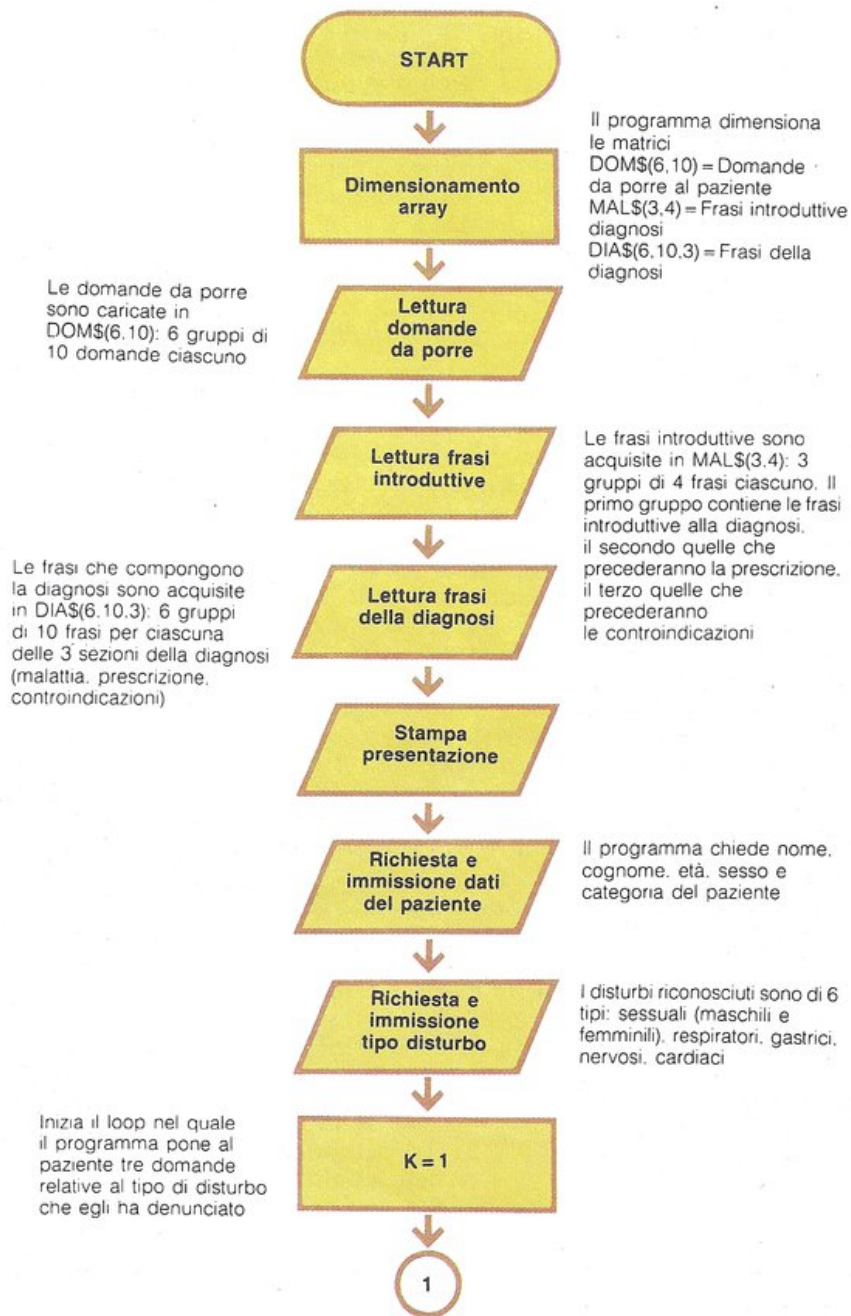
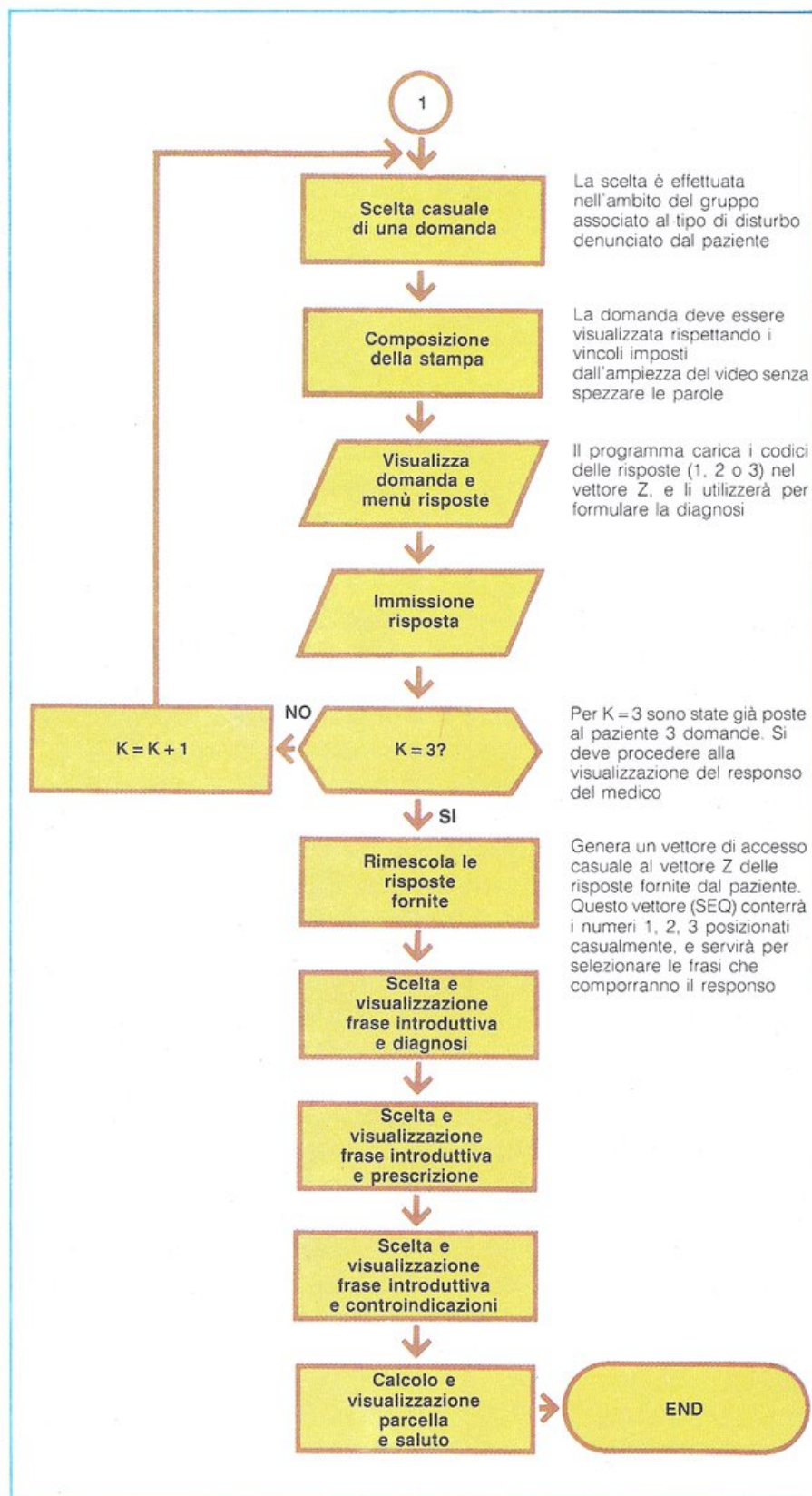
Prescrizione

1 / Scelta casuale di una frase introduttiva dalla seconda riga di MAL$
2 / Scelta di una cura dalla seconda pagina di DIA$, in dipendenza dalla risposta fornita dal paziente
3 / Stampa delle frasi selezionate

Controindicazioni

1 / Scelta casuale di una frase introduttiva dalla terza riga di MAL$
2 / Scelta di una controindicazione dalla terza pagina di DIA$, in dipendenza dalla risposta fornita dal paziente
3 / Stampa delle frasi selezionate.

Diagrammi di flusso

Il programma carica inizialmente le matrici DOM$, MAL$ e DIA$, quindi si presenta sul video e chiede al paziente-giocatore i suoi dati anagrafici e i disturbi che accusa. Nel ciclo successivo, in funzione del tipo dei disturbi denunciati, estrae da DOM$ tre domande a caso, le compone per la stampa, le visualizza e acquisisce le risposte. Successivamente rimescola (in un vettore di servizio) gli indici delle domande poste, in modo da fornire diagnosi diverse anche se il paziente immette risposte uguali a domande uguali. Infine costruisce le tre sezioni della diagnosi finale; in particolare, per ogni sezione il programma estrae casualmente una frase introduttiva ed una o due frasi per la costruzione della sezione del responso. L'ultimo blocco riguarda il calcolo della parcella e la sua visualizzazione insieme ad un messaggio di saluto.

È opportuno chiarire la logica di selezione delle frasi.
Il programma ha memorizzato in PD gli indici di colonna delle domande poste al paziente (l'indice di riga, che individua il gruppo, è dato dal tipo di disturbo che il paziente ha denunciato, ed eventualmente dal sesso dichiarato). Le risposte fornite dal paziente sono invece memorizzate, come codici numerici, in Z.
Per ciascuna sezione del responso il programma sceglierà casualmente una frase introduttiva dalla corrispondente riga di MAL$ e la caricherà nel buffer di stampa BF$. Quindi esaminerà una delle tre risposte fornite dal paziente (scelta a caso):
— se la risposta è SI visualizzerà la frase di DIA$ che occupa la posizione corrispondente a quella che occupa la domanda in DOM$;
— se la risposta è NO sceglierà a caso una frase all'interno di DIA$ nel gruppo che corrisponde al disturbo accusato;
— se la risposta è POCO/TALVOLTA visualizzerà due frasi: la prima è scelta a caso nel gruppo di DIA$ che corrisponde al disturbo accusato; la seconda è la frase di DIA$ che occupa la posizione corrispondente a quella occupata dalla domanda in DOM$.

Per la composizione e per la stampa delle righe conviene utilizzare una subroutine apposita, che può avere la struttura visibile qui a lato.
Nel primo blocco sono caricati in BF$ (buffer di stampa) la frase appena estratta ed il residuo eventualmente rimasto nel buffer; quindi, dopo aver atteso uno spazio da tastiera per continuare, si controlla che il contenuto del buffer entri in una sola riga del video.
In caso affermativo BF$ è stampato, altrimenti si entra in un loop che determina il punto in cui si può suddividere il buffer cercando il primo spazio che abbia a sinistra meno di 40 caratteri. Chiamata con SC la posizione in cui esiste lo spazio, si può stampare il buffer fino al carattere SC – 1 e quindi si salva in testa a BF$ tutto ciò che è rimasto, cioè i caratteri da SC in poi.
Si ripete poi la verifica su BF$ e lo si stampa fino a lasciarvi un numero di caratteri minore di 40.

Il programma

```
100 REM ****************************
110 REM * UN MEDICO POCO SERIO *
120 REM ****************************
140 REM VARIABILI UTILIZZATE
160 REM DOM$()     : MATRICE DELLE DOMANDE
170 REM MAL$()     : MATRICE FRASI INTRODUZIONE DIAGNOSI
180 REM DIA$()     : MATRICE FRASI DIAGNOSTICHE
190 REM I,J,K      : INDICI DI LOOP
200 REM N$         : NOME
210 REM C$         : COGNOME
220 REM ET         : ETA'
230 REM S$         : SESSO
240 REM S          : CODICE SESSO
250 REM PR         : CODICE PROFESSIONE
260 REM DS         : CODICE DISTURBO
270 REM X()        : ARRAY DI NUMERI CASUALI
280 REM X1         : NUMERO CASUALE
290 REM PD()       : ARRAY CONTENENTE LE POSIZIONI DELLE DOMANDE SCELTE
310 REM Z()        : ARRAY CONTENENTE I CODICI DELLE RISPOSTE
320 REM RP         : NUMERO CASUALE PRECEDENTE
330 REM BF$        : BUFFER DI STAMPA
340 REM SC         : VARIABILE DI SCANSIONE IN BF$
350 REM PA         : PARCELLA
360 REM RIT        : CONTATORE LOOP DI RITARDO
370 REM SEQ()      : VETTORE SEQUENZA DOMANDE
380 REM AUS        : VARIABILE DI APPOGGIO
381 REM PG$        : NOME PROGRAMMA
400 REM CARATTERI DI CONTROLLO
420 REM CHR$(18) : RVS/ON "█"
430 REM CHR$(147): CLR =  "♥"
450 REM DIMENSIONAMENTO MATRICI
470 DIM DOM$(6,10),X(3),Z(3)
480 DIM DIA$(6,10,3),PD(3)
490 DIM MAL$(3,4),SEQ(3)
510 REM  LETTURA DELLE DOMANDE
530 FOR I=1 TO 6
540 FOR J=1 TO 10
550 READ DOM$(I,J)
560 NEXT J
570 NEXT I
590 REM  LETTURA FRASI INTRODUTTIVE DIAGNOSI
610 FOR I=1 TO 3
620 FOR J=1 TO 4
630 READ MAL$(I,J)
640 NEXT J
650 NEXT I
654 REM  LETTURA FRASI RESPONSO
660 FOR K=1 TO 3
670 FOR I=1 TO 6
680 FOR J=1 TO 10
690 READ DIA$(I,J,K)
700 NEXT J,I,K
720 REM  PRESENTAZIONE
731 PG$="UN MEDICO POCO SERIO":GOSUB 62000
740 PRINT CHR$(147);
750 PRINT CHR$(18);
760 PRINT TAB(8);" UN MEDICO POCO SERIO "
770 PRINT
780 PRINT " BUONGIORNO,SI SIEDA"
790 PRINT " E RIEMPIA LA SCHEDA"
800 PRINT
820 REM ACQUISIZIONE DATI CLINICI E CONTROLLI
840 INPUT" NOME        ";N$
850 IF N$="" THEN PRINT CHR$(145);:GOTO 840
860 INPUT" COGNOME    ";C$
870 IF C$="" THEN PRINT CHR$(145);:GOTO 860
880 INPUT" ETA'        ";ET
890 IF ET<=0 THEN PRINT CHR$(145);:GOTO 880
900 INPUT " SESSO(M/F)";S$
910 IF S$ <>"M" AND S$<>"F" THEN PRINT CHR$(145);:GOTO 900
920 PRINT CHR$(147)
930 PRINT "  IN QUALE DI"
940 PRINT "  QUESTE CATEGORIE"
950 PRINT "  SI RICONOSCE?"
960 PRINT
970 PRINT " 1) DISOCCUPATO"
980 PRINT " 2) STUDENTE"
990 PRINT " 3) LIBERO PROFESS."
1000 PRINT" 4) DIPENDENTE"
1010 PRINT" 5) CASALINGA/O"
1020 PRINT
1030 INPUT"    QUALE";PR
1040 IF PR <1 OR PR > 5 THEN PRINT CHR$(145);:GOTO 1030
1050 PRINT CHR$(147)
1060 PRINT "  DI CHE GENERE DI"
1070 PRINT "  DISTURBI SOFFRE?"
1080 PRINT
1090 PRINT " 1) SESSUALI"
1100 PRINT " 2) RESPIRATORI"
1110 PRINT " 3) GASTRICI"
```

Il programma inizia con le istruzioni 470÷490 nelle quali vengono dimensionate le matrici usate per le frasi. Le frasi sono memorizzate nelle matrici alle linee 530÷700 con istruzioni READ, che leggono i DATA che si trovano a partire dalla linea 2630.
Nelle 731÷800 il programma si presenta (routine 62000) e quindi (linee 840÷1160) chiede al giocatore i dati anagrafici e i disturbi accusati; tutti gli input sono controllati, in modo da non permettere l'introduzione di stringhe nulle o fuori dei valori consentiti (linee 850, 870, 890, 910, 1040, 1160).

Nelle 1170, 1180 il programma trasforma il sesso in un indice numerico S, che vale 1 per il sesso maschile e 2 per quello femminile. In 1190, 1200 inizializza DS (che individua il blocco di domande e/o frasi in funzione del tipo di disturbo) con un valore che tiene conto del sesso del paziente. Nel ciclo 1250÷1640 pone al giocatore tre domande scelte a caso nel blocco corrispondente ai disturbi accusati. In particolare, alla linea 1270 carica in PD un numero casuale tra 1 e 10 che corrisponde ad una delle dieci possibili domande, controlla se la domanda è già stata fatta (linee 1330÷1350), la visualizza (linee 1390÷1470), assieme al menù delle possibili risposte (linee 1520÷1560), ed infine legge la risposta scelta (linea 1570), controllando che sia ammissibile (linea 1580), e la pone in Z. Nelle istruzioni 1690÷2190 è costruito e visualizzato il responso. Le linee 1690÷1800 caricano nel vettore di comodo SEQ i numeri 1, 2, 3 e li rimescolano casualmente. Gli elementi di SEQ, estratti sequenzialmente uno alla volta, costituiranno i puntatori agli elementi dei vettori delle risposte (Z) e delle domande poste (PD). Alla linea 1870 inizia il loop di scansione del vettore SEQ, nel quale è inserita la fase di selezione delle frasi che compongono il responso. Il ciclo è controllato dall'indice K: per K=1 è visualizzata la prima sezione del responso (diagnosi), per K=2 la seconda (prescrizione) e per K=3 la terza

```
1120 PRINT " 4) NERVOSI"
1130 PRINT " 5) CARDIACI"
1140 PRINT
1150 INPUT "     QUALE";DS
1160 IF DS<1 OR DS> 5 THEN PRINT CHR$(145);:GOTO 1150
1170 IF S$="M" THEN LET S=1:GOTO 1190
1180 LET S =2
1190 IF DS =1 THEN LET I=S:GOTO 1250
1200 LET I=DS+1
1220 REM SCELTA DI TRE DOMANDE CASUALI
1230 REM RELATIVE AL TIPO DI DISTURBO
1250 FOR K=1 TO 3
1260 PRINT CHR$(147)
1270 PD(K)= INT (RND(0)*10)+1
1290 REM CONTROLLA CHE LA DOMANDA NON
1300 REM SIA STATA GIA' FORMULATA
1310 REM IN CASO DI RISPOSTA POSITIVA RIPETE LA SCELTA
1330 FOR J=K-1 TO 1 STEP -1
1340 IF PD(J)=PD(K) THEN GOTO 1270
1350 NEXT J
1370 REM  PREPARA LA STAMPA DELLA DOMANDA
1390 LET BF$=DOM$(I,PD(K))
1410 REM CHIAMA LA SUBROUTINE DI COMPOSIZIONE
1430 GOSUB 2510
1450 REM  STAMPA L'ULTIMA RIGA DELLA DOMANDA
1470 PRINT BF$
1520 PRINT
1530 PRINT "  1)SI"
1540 PRINT "  2)NO"
1550 PRINT "  3)POCO/TALVOLTA
1560 PRINT
1570 INPUT "     EBBENE";Z(K)
1580 IF Z(K)<1 OR Z(K)>3 THEN PRINT CHR$(145);:GOTO 1570
1590 PRINT
1610 REM  RITARDO
1630 GOSUB 2440
1640 NEXT K
1650 PRINT CHR$(147)
1670 REM  INIZIALIZZAZIONE VETTORE SEQUENZA DOMANDE
1690 FOR J=1 TO 3
1700 LET SEQ(J)=J
1710 NEXT J
1730 REM  INVERSIONE CASUALE SEQUENZA DOMANDE
1750 FOR J=1 TO 3
1760 LET X1=INT(RND(0)*3)+1
1770 LET AUS=SEQ(X1)
1780 LET SEQ(X1)=SEQ(J)
1790 LET SEQ(J)=AUS
1800 NEXT J
1804 REM  AZZERA IL CONTENUTO DEL BUFFER PRECEDENTE
1808 LET BF$=""
1820 REM  LOOP DI DIAGNOSI SU TRE PASSAGGI:
1830 REM  IL PRIMO ASSEGNA UNA FRASE INTRODUTTIVA E UNA MALATTIA
1840 REM  IL SECONDO UNA FRASE INTRODUTTIVA E UNA PRESCRIZIONE
1850 REM  IL TERZO UNAFRASE INTRODUTTIVA E UNA CONTROINDICAZIONE
1870 FOR K=1 TO 3
1890 REM  SCELTA DI UNA FRASE INTRODUTTIVA CASUALE
1910 LET X(K)=INT(RND(0)*4)+1
1920 LET BF$=BF$+MAL$(K,X(K))+" "
1940 REM  RITARDO
1960 GOSUB 2440
1980 REM  SCELTA DI UNA O PIU' FRASI DIAGNOSTICHE
1990 REM  IN RELAZIONE ALLE RISPOSTE DATE
2010 IF Z(SEQ(K))=1 THEN LET BF$=BF$+DIA$(I,PD(SEQ(K)),K)+".":GOTO 2100
2020 IF Z(SEQ(K))=2 THEN LET BF$=BF$+DIA$(I,RND(0)*10+1,K)+".":GOTO 2100
2030 LET X1=INT(RND(0)*10)+1
2040 IF X1=PD(SEQ(K)) THEN GOTO 2030
2050 LET BF$=BF$+DIA$(I,X1,K)+" E "
2060 LET BF$=BF$+DIA$(I,PD(SEQ(K)),K) + "."
2080 REM  RITARDO
2100 GOSUB 2440
2120 REM   CHIAMA LA SUBROUTINE DI COMPOSIZIONE
2140 GOSUB 2510
2150 NEXT K
2170 REM  STAMPA IL CONTENUTO RIMANENTE DEL BUFFER
2190 PRINT BF$
2210 REM  CALCOLO DELLA PARCELLA
2230 LET PA = (6-DS+INT(RND(0)*5))*10000
2240 PRINT
2260 REM  FRASI CONCLUSIVE CONDIZIONATE
2270 REM  DALL'ESITO DELLA DIAGNOSI
2290 IF X(1)=4 THEN PRINT "ADDIO,";: GOTO2320
2300 PRINT "ARRIVEDERCI A PRESTO,";
2310 PRINT
2320 PRINT "SI PUO' ACCOMODARE"
2330 PRINT "DALLA MIA SEGRETARIA."
2340 PRINT
2360 REM  RITARDO
2380 GOSUB 2440
2390 PRINT "MI DEVE L. ";PA
```

(controindicazioni). Fissato il valore di K (cioè la sezione del responso e quindi la pagina di DIA$), le linee 1910, 1920 caricano nel buffer di stampa BF$ una frase introduttiva scelta a caso nella riga di MAL$ corrispondente a quella sezione.
Subito dopo si estrae il K-esimo valore contenuto in SEQ (che sarà 1, 2 o 3) e con esso si entra nel vettore delle risposte (Z).
— Se la risposta a cui si accede è SI (linea 2010) il programma accoda a BF$ la frase (del K-esimo piano di DIA$) che corrisponde idealmente alla domanda posta dal computer. Questa frase ha indici I, PD (SEQ(K)), K.
— Se la risposta è NO (linea 2020), cioè se Z(SEQ(K))=2, il programma sceglie una frase a caso (dal K-esimo piano di DIA$) nell'ambito della riga dedicata al disturbo dichiarato (I).
— Se la risposta è POCO/TALVOLTA (linee 2040÷2060) il programma sceglie due frasi: la prima (linee 2030÷2050) casualmente nell'I-esima riga (K-esima pagina) di DIA$, la seconda come se la risposta fosse stata SI.
Dopo la composizione di una data sezione del responso (in generale la K-esima) il programma entra in un loop di attesa (subroutine 2440), dopodiché chiama la subroutine di stampa (linea 2140) e passa al successivo valore di K (linea 2150).
Uscito dal ciclo di generazione del responso nelle 2230÷2390 il programma calcola la parcella e visualizza un messaggio di saluto. Da notare (linee 2290÷2310) che di

```
2395 FOR RIT=1 TO 6000 : NEXT RIT
2400 PRINT"[CLR]":POKE 53280,14:POKE 53281,6:END
2420 REM  SUBROUTINE DI RITARDO CASUALE
2440 LET X1=RND (0) * 4
2450 FOR  RIT=1 TO X1*500
2460 NEXT RIT
2470 RETURN
2490 REM  SUBROUTINE DI COMPOSIZIONE STAMPA
2510 LET BF$=BF$+" "
2520 IF LEN (BF$) < 40 THEN RETURN
2530 LET SC = 40
2540 IF MID$ (BF$,SC,1)= " " THEN GOTO 2570
2550 LET SC = SC-1
2560 GOTO 2540
2570 PRINT LEFT$(BF$,SC-1)
2580 LET BF$=RIGHT$(BF$,LEN(BF$)-SC)
2590 GOTO 2520
2610 REM  DOMANDE DISTURBI SESSUALI MASCHILI
2630 DATA AVVERTE UN FORTE PRURITO AI GENITALI?
2640 DATA LE DONNE LE FANNO LETTERALMENTE PERDERE LA TESTA?
2650 DATA E' UN TIPO POCO SENSUALE?
2660 DATA SECONDO LEI ALAIN DELON E' UN BELL'UOMO?
2670 DATA LE PIACE GUARDARSI ALLO SPECCHIO?
2680 DATA HA AVUTO GLI ORECCHIONI DOPO LA PUBERTA'?
2690 DATA MANGIA LE PAPAJE?
2700 DATA AMA SUA MADRE?
2710 DATA ACCUSA DOLORI ALLA REGIONE SUBINGUINALE?
2720 DATA FA USO DI DROGHE?
2740 REM  DOMANDE DISTURBI SESSUALI FEMMINILI
2760 DATA RAGGIUNGE L'ORGASMO?
2770 DATA GLI UOMINI LE FANNO PERDERE LETTERALMENTE LA TESTA?
2780 DATA AVVERTE FORTI PRURITI ALLA ZONA SUBINGUINALE?
2790 DATA HA LA VOCE MOLTO BASSA?
2800 DATA RISCONTRA LA PRESENZA DI PIAGHE SULLA PELLE?
2810 DATA HA ANIMALI IN CASA?
2820 DATA HA PAURA DEI LADRI?
2830 DATA TEME LE MALATTIE?
2840 DATA AVVERTE DOLORI ALLA REGIONE OVARICA?
2850 DATA HA UN CICLO REGOLARE?
2870 REM  DOMANDE MALATTIE RESPIRATORIE
2890 DATA HA DIFFICOLTA' NELLA DEGLUTIZIONE?
2900 DATA STARNUTISCE SPESSO?
2910 DATA HA DEI BUBBONI SOTTO LE ASCELLE?
2920 DATA HA UN DOLORE ALLA FRONTE?
2930 DATA RESPIRA A VOLTE CON DIFFICOLTA'
2940 DATA TOSSISCE SENZA CONTROLLO?
2950 DATA HA IL PALATO E LA LINGUA ARROSSATI?
2960 DATA HA UNA FORTE FEBBRE CON BRIVIDI?
2970 DATA HA FORTI DOLORI AL TORACE?
2980 DATA HA I MUSCOLI STERNOCLEIDOMASTOIDEI SPORGENTI?
3000 REM  DOMANDE DISTURBI GASTRICI
3020 DATA   VOMITA SPESSO?
3030 DATA HA UN FORTE DOLORE ALL'INGUINE?
3040 DATA AVVERTE BRUCIORI ALLO STOMACO?
3050 DATA AVVERTE DOLORI AL FEGATO?
3060 DATA FA IL BAGNO NEL TEVERE?
3070 DATA SENTE UN FORMICOLIO QUANDO SI SIEDE?
3080 DATA LE PIACE IL VERMOUTH?
3090 DATA MANGIA MOLTI DOLCI?
3100 DATA HA BEVUTO LATTE NON STERILIZZATO?
3110 DATA HA FATTO VIAGGI IN NORD AFRICA?
3130 REM  DOMANDE DISTURBI NERVOSI
3150 DATA HA PAURA DEL PROSSIMO?
3160 DATA LE TREMANO LE MANI?
3170 DATA CONDUCE UNA VITA STRESSANTE?
3180 DATA RACCONTA SPESSO FROTTOLE?
3190 DATA SI IRRITA SPESSO?
3200 DATA AVVERTE SPASMI MUSCOLARI CASUALI?
3210 DATA RUBA MAI AL SUPERMARKET?
3220 DATA ODIA GLI ASCENSORI?
3230 DATA VEDE I FILMS DI MARCO FERRERI?
3240 DATA LE PIACE IL COLORE VERDE?
3260 REM  DOMANDE DISTURBI CARDIACI
3280 DATA HA DIFFICOLTA' NELLA DEAMBULAZIONE?
3290 DATA IL CUORE LE BATTE COME UN MARTELLO PNEUMATICO?
3300 DATA HA UN FORTE DOLORE RETROSTERNALE?
3310 DATA SI SENTE DEBOLE?
3320 DATA HA FREQUENTI GIRAMENTI DI TESTA?
3330 DATA LAVORA MOLTO?
3340 DATA LE CAPITA DI DIRE COSE SENSA SENSO?
3350 DATA HA GIA' FATTO TESTAMENTO?
3360 DATA PRENDE LE MEDICINE CON L'INVOLUCRO?
3370 DATA HA DIFFICOLTA' A FARE LUNGHE CORSE?
3390 REM  FRASI INTRODUTTIVE DIAGNOSI MALATTIA
3410 DATA LA SUA MALATTIA E'
3420 DATA CERTAMENTE LEI SOFFRE DI
3430 DATA CREDO PROPRIO CHE SOFFRA DI
3440 DATA LEI STA PER MORIRE DI
3460 REM  FRASI INTRODUTTIVE PRESCRIZIONE
3480 DATA LE CONSIGLIO DI
```

solito il messaggio è ARRIVEDERCI, mentre se è stata diagnosticata la morte imminente il messaggio è... ADDIO. Notiamo infine che la subroutine 2440, chiamata in più punti del programma, genera un ritardo non costante. Nelle istruzioni DATA da 2630 a 5920 sono caricate tutte le domande e le frasi utilizzate. In coda al programma si trovano le due solite routines per l'inializzazione delle costanti (linea 61000) e per la stampa del titolo (linea 62000).

```
3490 DATA HA BISOGNO DI
3500 DATA DOVREBBE
3510 DATA E' INDISPENSABILE
3530 REM  FRASI INTRODUTTIVE CONTROINDICAZIONI
3550 DATA D'ORA IN POI NON DEVE
3560 DATA LE SCONSIGLIO DI
3570 DATA NON DOVREBBE PIU'
3580 DATA SI ASTENGA DAL
3600 REM  MALATTIE SESSUALI MASCHILI
3620 DATA PARASSITOSI DA INSETTI
3630 DATA SATIRIASI
3640 DATA IMPOTENZA
3650 DATA TENDENZE OMOSESSUALI PROVOCATE DA UN TRAUMA INFANTILE
3660 DATA NARCISISMO
3670 DATA STERILITA' MOLTO PROBABILE
3680 DATA URETRITE
3690 DATA MISOGINIA
3700 DATA ORCHITE
3710 DATA TOSSICODIPENDENZA IN STATO AVANZATO
3730 REM  MALATTIE SESSUALI FEMMINILI
3750 DATA FRIGIDITA'
3760 DATA NINFOMANIA
3770 DATA PARASSITOSI
3780 DATA GINANDRIA
3790 DATA SIFILIDE
3800 DATA OVARITE
3810 DATA TURBE PSICHICHE
3820 DATA IPOCONDRIA
3830 DATA LEUCORREA
3840 DATA ORZAIOLO
3860 REM  MALATTIE RESPIRATORIE
3880 DATA FARINGITE
3890 DATA RINITE
3900 DATA VAIOLO
3910 DATA SINUSITE
3920 DATA ASMA
3930 DATA PERTOSSE
3940 DATA STOMATITE
3950 DATA POLMONITE
3960 DATA PLEURITE
3970 DATA ENFISEMA POLMONARE
3990 REM  MALATTIE GASTRICHE
4010 DATA GASTRITE
4020 DATA APPENDICITE
4030 DATA CALCOLI ALLA CISTIFELLEA
4040 DATA CALCOLI AL FEGATO
4050 DATA LEPTOSPIROSI
4060 DATA VERMICULITE
4070 DATA CIRROSI EPATICA
4080 DATA DIABETE GRASSO
4090 DATA SCORBUTO
4100 DATA EPATITE VIRALE
4120 REM  MALATTIE NERVOSE
4140 DATA SCHIZOFRENIA
4150 DATA MORBO DI PARKINSON
4160 DATA ESAURIMENTO NERVOSO
4170 DATA MITOMANIA PARANOIDE
4180 DATA ISTERISMO
4190 DATA EPILESSIA
4200 DATA CLEPTOMANIA
4210 DATA CLAUSTROFOBIA
4220 DATA DEPRESSIONE CRONICA
4230 DATA PAZZIA FURIOSA
4250 REM  MALATTIE CARDIACHE
4270 DATA VENE VARICOSE
4280 DATA TACHICARDIA
4290 DATA PERICARDITE
4300 DATA ANEMIA PERNICIOSA
4310 DATA PRESSIONE BASSA
4320 DATA IPERTENSIONE
4330 DATA ARTERIOSCLEROSI
4340 DATA INFARTO
4350 DATA TIFLITE
4360 DATA SOFFIO AL CUORE
4380 REM  RIMEDI MALATTIE SESSUALI MASCHILI
4400 DATA USARE ANTIPARASSITARI
4410 DATA BERE DECOTTI DI VALERIANA
4420 DATA FARE USO DI AFRODISIACI
4430 DATA FREQUENTARE UNO PSICOANALISTA
4440 DATA FARSI MOLTI AMICI
4450 DATA SPERARE IN UN MIRACOLO (ANDARE A LOURDES)
4460 DATA NON RIMANERE MAI SOLO
4470 DATA RISALIRE ALLE CAUSE DEL TRAUMA
4480 DATA ASTENERSI DAI RAPPORTI (PROVARE CON LE MOLTIPLICAZIONI)
4490 DATA EVITARE GLI STUPEFACENTI
4510 REM  RIMEDI MALATTIE SESSUALI FEMMINILI
4530 DATA ASSUMERE AFRODISIACI
4540 DATA FARE USO DI CALMANTI
4550 DATA FARE LAVANDE ANTIPARASSITARIE
4560 DATA SEGUIRE UNA CURA A BASE DI ORMONI FEMMINILI
```

```
4570 DATA EVITARE I RAPPORTI (PROVARE CON LE MOLTIPLICAZIONI)
4580 DATA MANGIARE BANANE FINO ALLA NAUSEA (DELLA TENIA)
4590 DATA ANDARE DA UNO PSICANALISTA
4600 DATA DARSI UNA CALMATA
4610 DATA MANGIARE DODICI UOVA SODE A SERA
4620 DATA FARE IMPACCHI DI ACQUA E TE'
4640 REM  RIMEDI MALATTIE RESPIRATORIE
4660 DATA FARE GARGARISMI TRE VOLTE AL GIORNO
4670 DATA INALARE ACIDO CITRICO CONCENTRATO
4680 DATA DISINFETTARE LA CASA
4690 DATA PRATICARE L'AGOPUNTURA
4700 DATA PRATICARE LA RESPIRAZIONE BOCCA A BOCCA
4710 DATA USARE SCIROPPO ESPETTORANTE
4720 DATA PRENDERE QUATTRO CAPSULE DI ANTIBIOTICO AL DI'
4730 DATA OSSERVARE RIPOSO ASSOLUTO PER UN MESE
4740 DATA FARE LA PUNTURA LOMBARE
4750 DATA OSSERVARE UNA DIETA POVERA DI GRASSI
4770 REM  RIMEDI MALATTIE GASTRICHE
4790 DATA MANGIARE BISTECCHE AI FERRI
4800 DATA OPERARSI AL PIU' PRESTO
4810 DATA CONDURRE UNA VITA PIU' SERENA
4820 DATA LAVORARE MENO
4830 DATA USARE ANTISETTICI
4840 DATA MASSAGGIARE LA ZONA INTERESSATA
4850 DATA DISINTOSSICARSI
4860 DATA DIMAGIRE DI 10 KG
4870 DATA FARE UNA CURA DI VITAMINA C
4880 DATA OSSERVARE UNA DIETA STRETTISSIMA
4900 REM  RIMEDI MALATTIE NERVOSE
4920 DATA STUDIARE FILOSOFIA ORIENTALE
4930 DATA SOTTOPORSI AD INTERVENTO CHIRURGICO
4940 DATA FARE UNA LUNGA VACANZA
4950 DATA RACCONTARE MENO FROTTOLE
4960 DATA BERE DECOTTI DI VALERIANA E INFUSI DI CAMOMILLA
4970 DATA CURARSI CON BARBITURICI
4980 DATA CERCARE DI ALLACCIARE MOLTE AMICIZIE
4990 DATA ANDARE DA UNO PSICHIATRA
5000 DATA ANDARE A VEDERE UN FILM COMICO
5010 DATA FARSI INTERNARE
5030 REM  RIMEDI MALATTIE CARDIACHE
5050 DATA MANGIARE PAPAJE
5060 DATA NON ECCITARSI PER UN NONNULLA
5070 DATA INIETTARSI PROGENIL-PAPAVERINA
5080 DATA FARE UNA BELLA MANGIATA
5090 DATA SEGUIRE UN REGIME ALIMENTARE ADEGUATO
5100 DATA USARE FARMACI DIURETICI
5110 DATA PRENDERE DECOTTI DI ORTICA
5120 DATA ANDARE IN VACANZA IN SCANDINAVIA
5130 DATA SOTTOPORSI AD OPERAZIONE CHIRURGICA
5140 DATA SOSTITUIRE LA VALVOLA MITRALICA
5160 REM  CONTROINDICAZIONI MALATTIE SESSUALI MASCHILI
5180 DATA TRASCURARE L'IGIENE INTIMA
5190 DATA AVERE RAPPORTI FREQUENTI
5200 DATA SOPPORTARE GROSSI SFORZI
5210 DATA PENSARCI TROPPO
5220 DATA GUARDARSI ALLO SPECCHIO
5230 DATA DESIDERARE FIGLI
5240 DATA LAVORARE TROPPO DI FANTASIA
5250 DATA PENSARE A SUA MADRE
5260 DATA MANGIARE AVOCADOS
5270 DATA ANDARE IN SUDAMERICA
5290 REM  CONTROINDICAZIONI MALATTIE SESSUALI FEMMINILI
5310 DATA SCOPRIRSI TROPPO
5320 DATA FREQUENTARE LE CASERME
5330 DATA TRASCURARE L'IGIENE INTIMA
5340 DATA USARE SCHIUMA DA BARBA
5350 DATA BERE LIMONATE
5360 DATA MANGIARE CIBI CRUDI
5370 DATA LEGGERE I LIBRI DI MORAVIA
5380 DATA FREQUENTARE I SELF-SERVICE
5390 DATA SOTTOVALUTARE LE PRESE DI CORRENTE
5400 DATA PORTARE SENZA RAGIONE LE DITA AGLI OCCHI
5420 REM  CONTROINDICAZIONI MALATTIE RESPIRATORIE
5440 DATA ANDARE AL MARE QUANDO FA FREDDO
5450 DATA USARE FAZZOLETTI POLVEROSI
5460 DATA FARE IL BAGNO NEL TEVERE
5470 DATA PENSARE INTENSAMENTE
5480 DATA PROVARE FORTI EMOZIONI
5490 DATA ANDARE IN MOTO QUANDO PIOVE
5500 DATA METTERE IN BOCCA TUTTO CIO' CHE CAPITA
5510 DATA PRENDERE FREDDO
5520 DATA ANDARE A SCIARE
5530 DATA FUMARE
5550 REM  CONTROINDICAZIONI MALATTIE GASTRICHE
5570 DATA MANGIARE PATATE FRITTE
5580 DATA MANGIARE PASTA
5590 DATA AVERE FORTI DIVERBI
5600 DATA FARE PRANZI LUCULLIANI
5610 DATA GIOCARE CON I RODITORI
5620 DATA ANDARE A CAVALLO
```

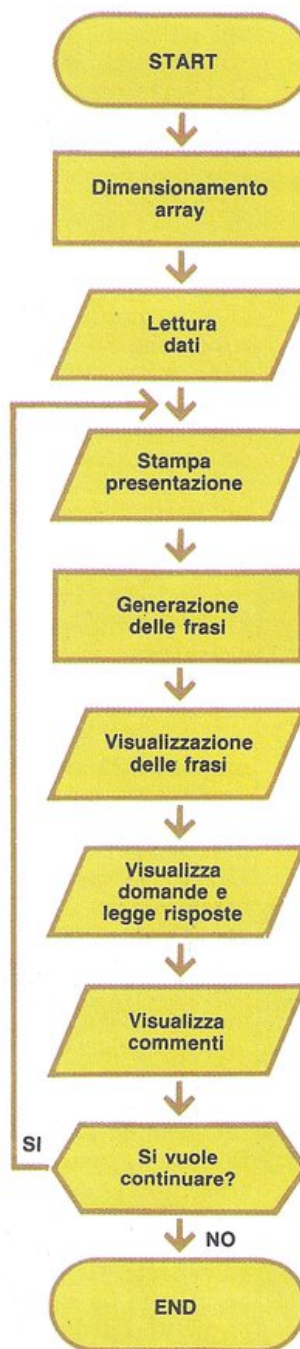
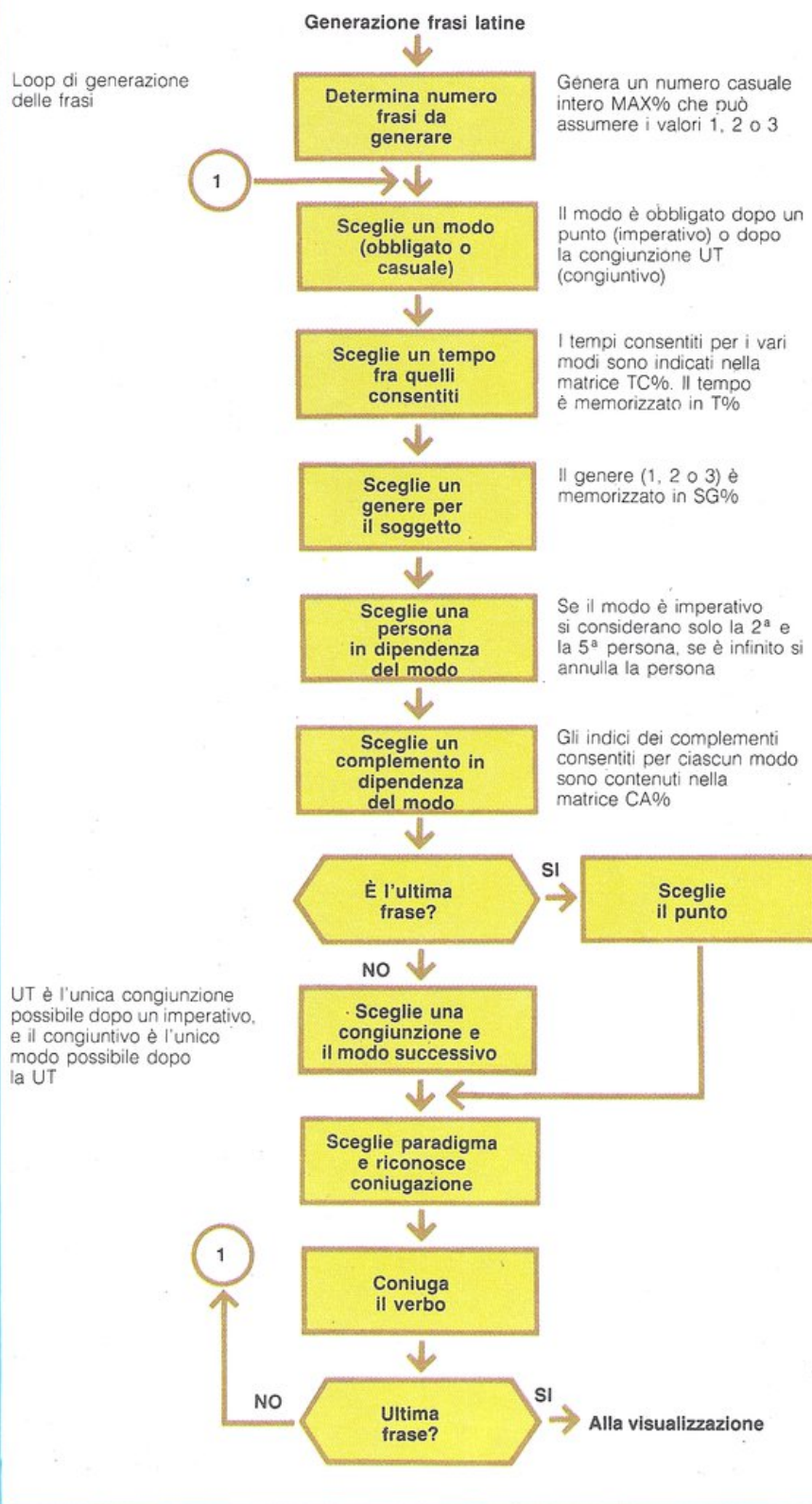
```
5630 DATA BERE ALCOOL
5640 DATA MANGIARE LA SACHER-TORTE
5650 DATA SEGUIRE DIETE POVERE DI FIBRE
5660 DATA ANDARE IN MAROCCO
5680 REM  CONTROINDICAZIONI MALATTIE NERVOSE
5700 DATA CIMENTARSI IN ATTIVITA' INTELLETTUALI TROPPO IMPEGNATIVE
5710 DATA BERE WHISKY
5720 DATA LAVORARE TROPPO
5730 DATA PARLARE A SPROPOSITO
5740 DATA GUIDARE L'AUTO
5750 DATA PROVARE VIOLENTE EMOZIONI
5760 DATA RUBARE OGGETTI...E LASCI STARE IL FERMACARTE
5770 DATA DORMIRE SUL BALCONE
5780 DATA FREQUENTARE PERSONE NOIOSE
5790 DATA ANDARE ALLO STADIO
5810 REM  CONTROINDICAZIONI MALATTIE CARDIACHE
5830 DATA CAMMINARE MOLTO
5840 DATA ASSUMERE DROGHE ECCITANTI
5850 DATA SALTARE I PASTI
5860 DATA ALZARSI PRESTO LA MATTINA
5870 DATA BERE CAFFE'
5880 DATA MANGIARE CIBI GRASSI
5890 DATA AVERE FORTI EMOZIONI
5900 DATA MANGIARE UOVA SODE CON IL GUSCIO
5910 DATA AFFATICARSI ECCESSIVAMENTE (EVITARE GLI ABUSI SESSUALI
5920 DATA FARE LUNGHE CORSE
60960 REM SUBROUTINE: INIZIALIZZAZIONE COSTANTI
60980 REM CIRCUITO VIDEO
61000 VI=53248
61020 REM CIRCUITO SUONO
61040 SI=54272
61060 REM MEMORIA VIDEO
61080 MV=1024
61100 REM MEMORIA COLORE
61120 MC=55296
61180 REM INIZIALIZZAZIONE CHIP SUONO
61200 FOR I=0 TO 24
61210 POKE SI+I,0
61220 NEXT I
61230 RETURN
61980 REM SUBROUTINE:STAMPA IL TITOLO DEL PROGRAMMA MEMORIZZATO IN PG$
62000 GOSUB 61000
62010 POKE VI+32,15
62020 POKE VI+33,15
62030 PRINT "█████";
62040 PRINT TAB(6);"┌──────────────────────────┐"
62050 FOR I=1 TO 5
62060 PRINT TAB(6);"|                          |"
62070 NEXT I
62080 PRINT TAB(6);"└──────────────────────────┘"
62090 PRINT TAB(6);"███████DIGITA ▓RETURN▒ PER PROSEGUIRE"
62100 PRINT "█████"
62110 FOR I=1 TO 5
62120 PRINT TAB(7);
62130 FOR J=1 TO 26
62140 PRINT "██ "
62150 NEXT J
62160 PRINT
62170 NEXT I
62190 REM ORA SCRIVO IL TITOLO
62210 PRINT "█████████";TAB((40-LEN(PG$))/2);"██";PG$
62220 GET Z9$
62230 IF Z9$<>CHR$(13) THEN GOTO 62220
62240 PRINT "██";
62250 RETURN
```

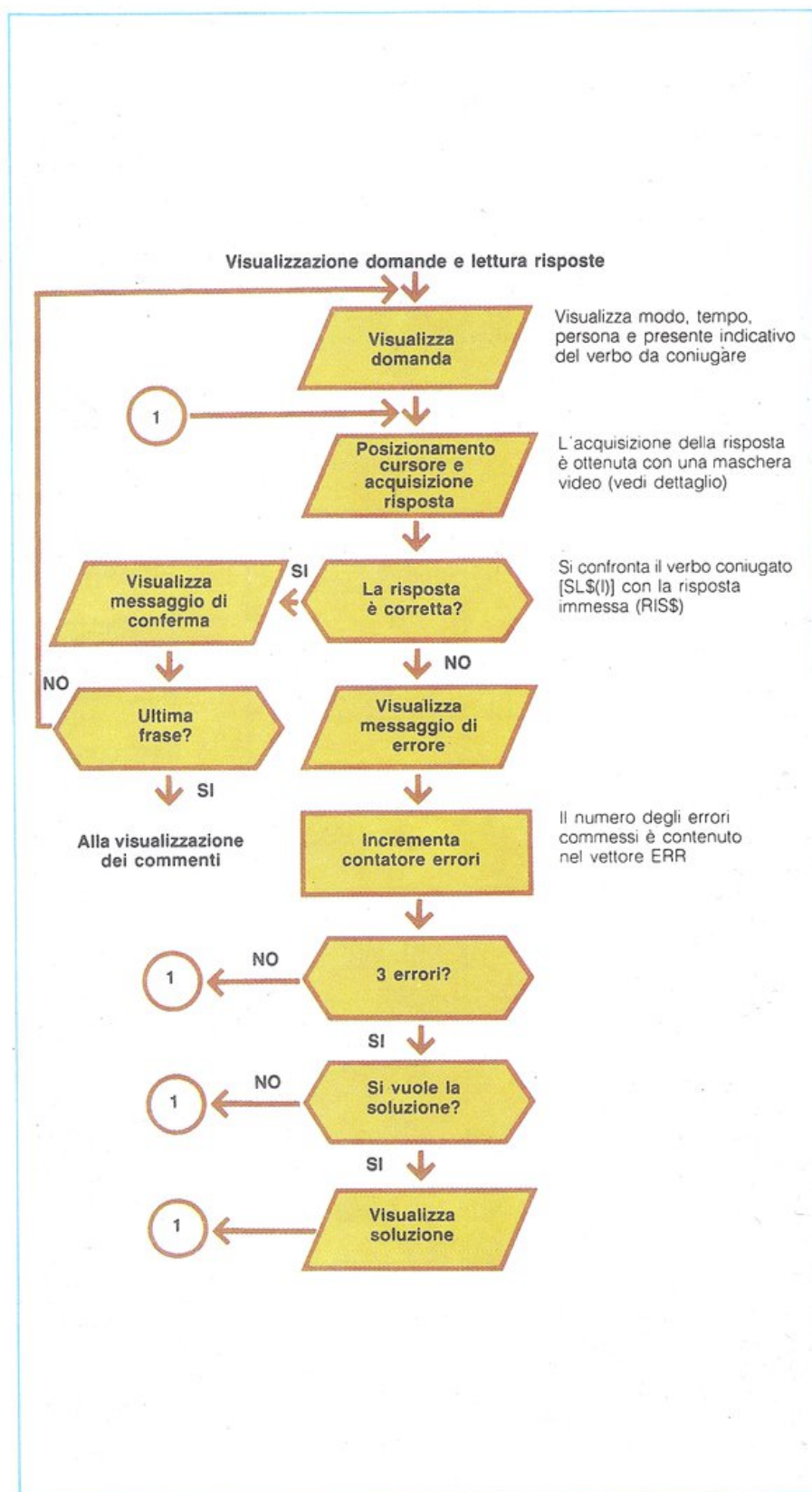
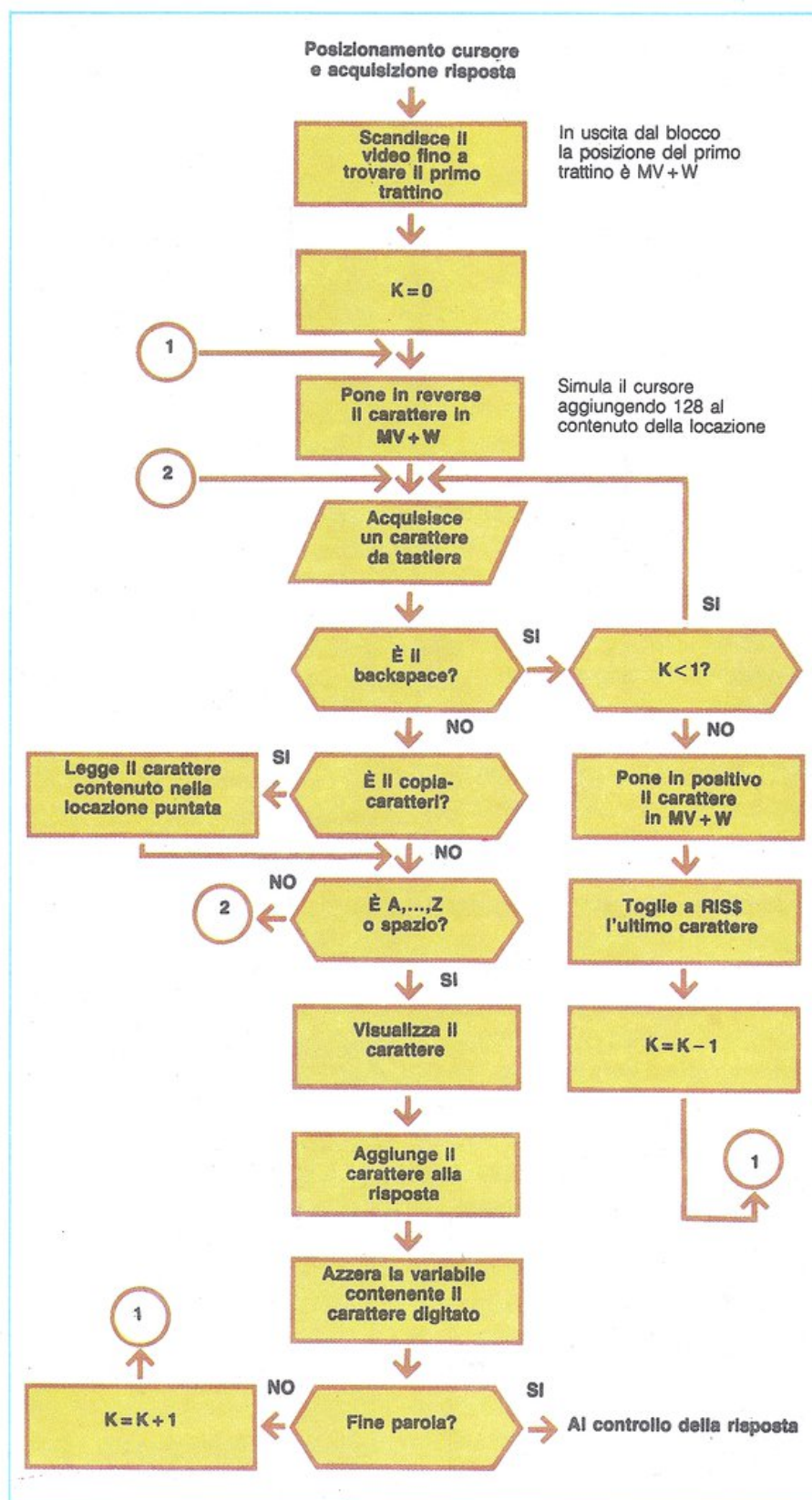
Computo, computas...

Il programma che ora considereremo consentirà all'utente di sottoporsi ad un esame sulla coniugazione dei verbi latini. Funzionerà presentando sul video un periodo in latino, nel quale i verbi sono tenuti nascosti e sono visualizzate in loro vece delle sequenze di trattini.
Dei verbi sono forniti il presente indicativo, il modo, il tempo e la persona, e si chiede all'utente di inserire nella frase i verbi coniugati. Il programma comunicherà poi se ciascuna coniugazione è esatta, fornendo eventualmente su richiesta la forma corretta. Si avranno infine la visualizzazione del numero di errori effettuati e gli aspetti della coniugazione dei verbi che il programma suggerirà di ripassare.

Analisi del problema

Il programma deve effettuare una generazione casuale di frasi latine, in modo che esse risultino sempre diverse. Tale generazione dovrà però essere eseguita rispettando le regole sintattiche e grammaticali della lingua latina, inserendo cioè nella frase i complementi richiesti dalla sua costruzione e declinando i sostantivi nel modo corretto.
La generazione avverrà decidendo dapprima in modo casuale il numero di proposizioni (tre al massimo) da comprendere nel periodo presentato all'utente e determinando poi (casualmente), per ogni proposizione, il modo ed il tempo del verbo compreso in essa.
I modi e i tempi considerati saranno elencati nei vettori MODI$ e TM$, ma, dato che i tempi tra i quali si può scegliere dipendono dal modo cui si riferiscono (per esempio, nel congiuntivo non esiste il futuro), sarà necessario introdurre una matrice TC% la quale, per ogni indice relativo ad un modo in MODI$, fornirà gli indici dei tempi in TM$ compatibili con il modo considerato. Sarà poi generato un soggetto a caso (a meno che il modo del verbo non sia l'infinito) scelto tra quelli contenuti nella matrice SO$, determinandone dapprima genere e persona, quindi il particolare sostantivo da usare al caso nominativo.
Dopo di ciò, il programma passerà alla generazione di un complemento: nel caso la frase sia al passivo ci dovrà essere un complemento d'agente, altrimenti sarà scelto uno degli altri tipi di complemento. La parola da inserire nella frase sarà scelta tra quelle elencate nel vettore CM$, dopo aver determinato il tipo del complemento compatibile con il modo usato per il verbo (la compatibilità tra modo del verbo e complementi è espressa dalla matrice CA%) e il genere e la persona del complemento, che saranno gli stessi già fissati per il soggetto.
Se la proposizione generata non è l'ultima del periodo, sarà scelta a caso una congiunzione e, dopo aver scelto un paradigma per il verbo ed avere coniugato il verbo nel modo, tempo e persona già scelti secondo tale paradigma, si passerà alla generazione della proposizione successiva.
Dopo aver generato il periodo, il programma lo presenterà su video sostituendo i verbi con sequenze di trattini, così da avere un trattino per ogni carattere del verbo coniugato. Il programma esaminerà successivamente un verbo alla volta, fornendone radice, modo, tempo e persona, ed aspetterà che l'utente scriva il verbo coniugato al posto dei trattini. Se quanto ha scritto l'utente è corretto si passa al verbo successivo, altrimenti il programma comunica che la risposta è sbagliata e ne attende una nuova. Dopo tre errori, se vuole, l'utente potrà farsi comunicare la risposta esatta dal programma.
Considerati tutti i verbi, il programma comunicherà il numero di errori effettuati, i modi e i tempi dei quali è stata sbagliata la coniugazione, dopo di che l'utente potrà specificare se vuole rieseguire l'intero programma generando un altro periodo o se preferisce terminare l'esecuzione.

Diagrammi di flusso

Il programma presenta una struttura abbastanza semplice: consiste infatti in un singolo ciclo, all'interno del quale un certo numero di frasi viene generato, presentato e le risposte dell'utente vengono esaminate.

Alcuni dei blocchi interni a questo ciclo sono presentati in dettaglio: in particolare, il blocco di generazione delle frasi consiste a sua volta in un ciclo, eseguito per ogni frase, all'interno del quale viene determinata la struttura della frase considerata.

Il blocco di visualizzazione domande e lettura risposte è espanso in un ciclo, eseguito per ogni frase, di visualizzazione della domanda e di lettura delle risposte. Dato che per ogni frase si possono operare più tentativi di risposta, all'interno del ciclo precedente ne è posto un altro, che viene eseguito per acquisire un singolo tentativo di risposta.

Infine, è presentata l'espansione del blocco di acquisizione della risposta (interno al precedente), che permette all'utente di scrivere la risposta ed eventualmente correggerla, offrendo così una piccola possibilità di editing di frasi, possibilità che i più volenterosi potrebbero pensare di ampliare definendo nuovi comandi di modifica del testo scritto.

Il programma

```
50 REM ***********************
60 REM * COMPUTO,COMPUTAS.. *
70 REM ***********************
110 REM PRINCIPALI VARIABILI UTILIZZATE
130 REM MODI$()      :MODI
140 REM TM$()        :TEMPI
145 REM SG$()        :GENERI
150 REM TC%()        :TEMPI CONSENTITI
160 REM SO$()        :SOGGETTI
170 REM CM$()        :COMPLEMENTI
180 REM CA%()        :COMPLEMENTI AMMESSI
190 REM CN$()        :CONGIUNZIONI
200 REM DES$()       :DESINENZE
210 REM PD$()        :PARADIGMI
220 REM DGN$()       :DESINENZE GEN/NUM
230 REM MAX%()       :NUMERO FRASI
240 REM CD%          :MODO OBBLIGATO
260 REM M%           :MODO
270 REM T%           :TEMPO
280 REM SR%          :PERSONA
290 REM SG%          :GENERE
300 REM VR%()        :PERSONE SCELTE
305 REM VG%()        :GENERI SCELTI
310 REM VM%()        :MODI SCELTI
320 REM VT%()        :TEMPI SCELTI
330 REM ST$()        :SOGGETTI SCELTI
332 REM CC$()        :COMPLEMENTI SCELTI
335 REM B()          :PARADIGMI SCELTI
350 REM NP           :TIPO COMPLEMENTO
360 REM CG%          :CODICE CONGIUNZIONE
370 REM CZ$()        :CONGIUNZIONI SCELTE
380 REM RP%()        :NUMERO PARADIGMA
385 REM SL$()        :VERBI CONIUGATI
390 REM PRE$         :PRES.
400 REM PER$         :PERF.
410 REM SUP$         :SUP.
420 REM INF$         :INF.
440 REM C            :CODICE CONIUGAZIONE
450 REM R1$,R2$      :RIGHE MATRICE DESINENZE
470 REM MV           :LOCAZIONE MEMORIA VIDEO
480 REM W            :LOCAZIONE INIZIO MASCHERA
490 REM BF$          :BUFFER DI STAMPA
500 REM TT$          :CARATTERE DIGITATO
510 REM RIS$         :RISPOSTA
520 REM ERR()        :ERRORI COMMESSI
530 REM PG$          :NOME PROGRAMMA
550 REM CARATTERI DI CONTROLLO
570 REM CHR$(147):CLR =              "▯"
580 REM CHR$(145):CURSORE IN ALTO =  "┐"
590 REM CHR$(18) :RVS ON =           "▩"
600 REM CHR$(19) :HOME =             "▩"
610 REM CHR$(157):CURSORE A SINISTRA="▌"
620 REM CHR$(29) :CURSORE A DESTRA = "▐"
640 REM DIMENSIONAMENTO VARIABILI
660 DIM TC%(9,7),SO$(3,2,3),CM$(3,3,2,3)
664 DIM MODI$(9),TM$(6),CN$(5),ERR(3)
666 DIM VR%(3),VM%(3),VT%(3),ST$(3),VG%(3)
668 DIM CC$(3),B(3),CZ$(3),SL$(3),SG$(3)
670 DIM CA%(9,4),DES$(14,6),PD$(10,4),DGN$(12)
690 REM LETTURA VETTORI/MATRICI
740 FOR I =1 TO 9
750 READ MODI$(I)
760 NEXT I
780 REM TEMPI
800 FOR I=1 TO 6
810 READ TM$(I)
820 NEXT I
840 REM MODI/TEMPI
860 FOR I=1 TO 9
870 FOR J=1 TO 7
880 READ TC%(I,J)
890 NEXT J,I
910 REM SOGGETTI
930 FOR I=1 TO 3
940 FOR J=0 TO 1
950 FOR K=1 TO 3
960 READ SO$(I,J,K)
970 NEXT K,J,I
990 REM COMPLEMENTI
1010 FOR I=1 TO 3
```

Le linee 660÷670 dimensionano gli array utilizzati nel seguito, che vengono poi letti con le istruzioni 690÷1390. Le linee 1421÷1450, chiamando la routine 62000, fanno scrivere sul video l'intestazione del programma: la linea 62000 chiama a sua volta la routine 61000, che inizializza un certo numero di costanti, oltre ai registri relativi al suono. La linea 1490 chiama la routine 3130, che genera le

frasi da presentare all'utente in modo casuale, pur rispettando le regole sintattiche di formazione delle frasi. In particolare, per ogni verbo della frase, alle linee 3200÷3240 viene scelto il modo e alle linee 3280÷3300 viene scelto, a caso, un tempo fra quelli esistenti nel modo già scelto. L'istruzione 3340 determina il genere del soggetto, mentre le linee 3490÷3611 scelgono la persona a cui declinare il verbo: se il modo scelto è però l'infinito, le linee 3619÷3640 eliminano il soggetto, creando una proposizione oggettiva. Le linee 3680÷3740 assegnano un complemento alla proposizione, mentre le 3780÷3980 decidono come chiudere la frase, se con una congiunzione o un punto. Le linee 4020÷4070 scelgono poi il verbo da declinare e le 4110÷4300 lo declinano nella forma già scelta chiamando la routine 5570. Usciti dalla routine, la linea 1530 chiama la routine 4340 che stampa la frase generata sul video. Quindi, la linea 1570 chiama la routine 4540 che, per ogni verbo contenuto nella frase, specifica tempo, modo e persona a cui va coniugato, legge le risposte dell'utente e tiene un conteggio del numero di errori effettuati. La linea 1610 chiama la routine 5350 che specifica il numero di errori commessi e indica quali parti dell'insieme di regole di coniugazione di verbi latini conviene che l'utente ripassi. Infine, le linee 1650÷1690 chiedono all'utente se vuole continuare: in caso affermativo si torna alla presentazione, altrimenti il programma termina.

```
1020 FOR J=1 TO 3
1030 FOR K=0 TO 1
1040 FOR Q=1 TO 3
1050 READ CM$(I,J,K,Q)
1060 NEXT Q,K,J,I
1080 REM COMPLEMENTI AMMESSI
1100 FOR I=1 TO 9
1110 FOR J=1 TO 4
1120 READ CA%(I,J)
1130 NEXT J,I
1150 REM CONGIUNZIONI
1170 FOR I=1 TO 5
1180 READ CN$(I)
1190 NEXT I
1210 REM DESINENZE
1230 FOR I =1 TO 14
1240 FOR J=1 TO 6
1250 READ DES$(I,J)
1260 NEXT J,I
1280 REM PARADIGMI
1300 FOR I=1 TO 10
1310 FOR J=1 TO 4
1320 READ PD$(I,J)
1330 NEXT J,I
1350 REM DESINENZE GEN/NUM
1370 FOR I=1 TO 12
1380 READ DGN$(I)
1390 NEXT I
1395 REM GENERI
1400 FOR I=1 TO 3
1405 READ SG$(I)
1407 NEXT I
1410 REM PRESENTAZIONE
1421 PG$="COMPUTO, COMPUTAS...":GOSUB 62000
1430 PRINT CHR$(147);
1440 PRINT CHR$(18);
1450 PRINT TAB(9);" COMPUTO,COMPUTAS... ":PRINT:PRINT
1470 REM GENERAZIONE FRASI
1490 GOSUB 3130
1510 REM STAMPA FRASI
1530 GOSUB 4340
1550 REM FORMULAZIONE DOMANDE
1570 GOSUB 4540
1590 REM STAMPA COMMENTI FINALI
1610 GOSUB 5350
1630 REM CONTINUAZIONE O FINE PROGRAMMA
1650 PRINT:PRINT" RICOMINCIAMO? (S/N)"
1660 GET TT$
1670 IF TT$="S" THEN GOTO 1430
1680 IF TT$="N" THEN POKE 53280,14:POKE 53281,6:PRINT"":PRINT CHR$(147):END
1690 GOTO 1660
1710 REM MODI
1730 DATA IND.ATT.
1740 DATA CONG.ATT.
1750 DATA IMPER.ATT.
1760 DATA INF.ATT.
1770 DATA IND.PASS.
1780 DATA CONG.PASS.
1790 DATA IMPER.PASS.
1800 DATA INF.PASS.
1810 DATA GERUNDIVO
1830 REM TEMPI
1850 DATA PRES.
1860 DATA IMPERF.
1870 DATA FUTURO
1880 DATA PERFETTO
1890 DATA PIUCCH.
1900 DATA FUT.ANT.
1920 REM COMBINAZIONI AMMESSE MODI/TEMPI
1940 DATA 6,1,2,3,4,5,6
1950 DATA 4,1,2,4,5,0,0
1960 DATA 1,1,0,0,0,0,0
1970 DATA 3,1,3,4,0,0,0
1980 DATA 6,1,2,3,4,5,6
1990 DATA 4,1,2,4,5,0,0
2000 DATA 0,0,0,0,0,0,0
2010 DATA 3,1,3,4,0,0,0
2020 DATA 1,1,0,0,0,0,0
2040 REM SOGGETTI MASCHILI
2060 DATA AFFECTUS,AMOR,NOVUS MOS
```

Il numero limitato di termini latini disponibili e la casualità con la quale sono legati tra loro determinano a volte la mancanza di un nesso logico fra le diverse frasi presentate dal computer. Il programma non può infatti sostituire un buon testo o un bravo professore: resta comunque una sfida ai «latinisti» di ieri e una verifica grammaticale per gli studenti di oggi.

```
2070 DATA AFFECTUS,HORRORES,MORES
2090 REM SOGGETTI FEMMINILI
2110 DATA SORS,MORS,PRUDENTIA
2120 DATA RES SECUNDAE,ANGUSTIAE,RELIGIONES
2140 REM SOGGETTI NEUTRI
2160 DATA FATUM,CONSILIUM,STUDIUM
2180 REM SOGGETTI NEUTRI
2200 DATA ODIA,SOLACIA,OFFICIA
2220 REM COMPLEMENTI OGGETTI
2250 REM MASCHILI
2270 DATA PLAUSUM,ERROREM,PECCATUM
2280 DATA SOPORES,AFFECTOS,AMORES
2300 REM FEMMINILI
2320 DATA AVIDITATEM,SUPERBIAM,OBLIVIONEM
2330 DATA RES ABSURDAS,PRIVATIONES,CALAMITATES
2350 REM NEUTRI
2370 DATA OBLECTAMENTUM,STULTE FACTUM,OTIUM
2380 DATA LOGICA,OMNIA,OFFICIA
2400 REM COMPLEMENTI VARI SINGOLARI
2420 DATA OBLECTAMENTO,ONERE,HORRORE
2430 DATA ANIMI MOTU, SORTE,PROPTER ODIUM
2440 DATA CUM AVIDITATE,OB MORTEM,IN SOLACIO
2460 REM COMPLEMENTI VARI PLURALI
2480 DATA AFFECTIBUS,AMORIBUS,ARTIBUS
2490 DATA HORRORIBUS,OBLIVIONIBUS,PECCATIS
2500 DATA PRIVATIONIBUS,PLAUSIS,INANIBUS
2520 REM COMPLEMENTI D'AGENTE E DI CAUSA EFFICIENTE
2540 DATA SAPIENTIA,CONSILIIS,STUDIO
2550 DATA SORTE,FATO,MORTE
2560 DATA ODIO,PRUDENTIA,ARTE
2570 DATA A CICERONE,A CAESARE,A SANTIPPE
2580 DATA A CATILINA,AB ALEXANDRO,AB OGYGIA
2590 DATA ONERE,A CLAUDIO,DEIS
2610 REM MATRICE COMPLEMENTI AMMESSI
2630 DATA 3,1,2,3
2640 DATA 3,1,2,3
2650 DATA 3,1,2,3
2660 DATA 3,1,2,3
2670 DATA 2,2,3,0
2680 DATA 2,2,3,0
2690 DATA 2,2,3,0
2700 DATA 2,2,3,0
2710 DATA 1,2,0,0
2730 REM CONGIUNZIONI
2750 DATA QUOD,SED,UT,CUM,.
2770 REM DESINENZE
2790 DATA M,S,T,MUS,TIS,NT
2800 DATA R,RIS,TUR,MUR,MINI,NTUR
2810 DATA O,AS,AT,AMUS,ATIS,ANT
2820 DATA EO,ES,ET,EMUS,ETIS,ENT
2830 DATA O,IS,IT,IMUS,ITIS,UNT
2840 DATA IO,IS,IT,IMUS,ITIS,IUNT
2850 DATA BO,BIS,BIT,BIMUS,BITIS,BUNT
2860 DATA BOR,BERIS,BITUR,BIMUR,BIMINI,BUNTUR
2870 DATA AM,ES,ET,EMUS,ETIS,ENT
2880 DATA AR,ERIS,ETUR,EMUR,EMINI,ENTUR
2890 DATA I,ISTI,IT,IMUS,ISTIS,ERUNT
2900 DATA ERO,ERIS,ERIT,ERIMUS,ERITIS,ERINT
2910 DATA SUM,ES,EST,SUMUS,ESTIS,SUNT
2920 DATA ERO,ERIS,ERIT,ERIMUS,ERITIS,ERUNT
2940 REM PARADIGMI
2960 DATA COERCEO,COERCUI,COERCITUM,COERCERE
2970 DATA EXCITO,EXCITAVI,EXCITATUM,EXCITARE
2980 DATA CAPIO,CEPI,CAPTUM,CAPERE
2990 DATA OSTENDO,OSTENDI,OSTENSUM,OSTENDERE
3000 DATA TEMPERO,TEMPERAVI,TEMPERATUM,TEMPERARE
3010 DATA IMPEDIO,IMPEDIVI,IMPEDITUM,IMPEDIRE
3020 DATA AGNOSCO,AGNOVI,AGNITUM,AGNOSCERE
3030 DATA ADMINISTRO,ADMINISTRAVI,ADMINISTRATUM,ADMINISTRARE
3040 DATA INDUCO,INDUXI,INDUCTUM,INDUCERE
3050 DATA VINCO,VICI,VICTUM,VINCERE
3070 REM DESINENZE GEN/NUM
3090 DATA US,A,UM,I,AE,A,UM,AM,UM,OS,AS,A
3095 REM GENERI
3100 DATA MASCHILE,FEMMINILE,NEUTRA
3110 REM GENERAZIONE FRASI
3130 CD%=0
3140 MAX%=RND(0)*3+1
3150 FOR I=1 TO MAX%
3155 F=0
```

```
3170 REM SCELTA DEL MODO
3200 IF CD%=0 THEN M%=RND(0)*9+1:GOTO 3240
3210 M%=RND(0)+.5
3220 IF M%=1 THEN M%=CD%:GOTO 3280
3230 M%=CD%+4:GOTO 3280
3240 IF M%=3 OR M%=7 THEN GOTO 3200
3260 REM SCELTA DEL TEMPO TRA QUELLI CONSENTITI
3280 IF TC%(M%,1)= 0 THEN GOTO 3200
3290 T%=TC%(M%,RND(0)*TC%(M%,1)+2)
3300 VM%(I)=M%:VT%(I)=T%
3320 REM SCELTA DEL GENERE PER IL SOGGETTO
3340 SG%=RND(0)*3+1
3342 IF M%<=4 THEN VG%(I)=0:GO TO 3490
3343 IF M%<>9 AND T%<=3 THEN VG%(I)=0:GO TO 3490
3344 VG%(I)=SG%
3460 REM SE IL MODO E' IMPERATIVO SCEGLIE LA SECONDA O LA
3470 REM QUINTA PERSONA E SALTA LA SCELTA DEL SOGGETTO
3490 IF M%<>3 AND M%<>7 THEN GOTO 3570
3500 SR%=RND(0)+.5
3510 IF SR%=1 THEN  SR%=2:GOTO 3530
3520 SR%=5
3530 ST$(I)="":GOTO 3640
3550 REM SCELTA DELLA PERSONA DEL SOGGETTO
3570 SR%=RND(0)*6+1:IF SR%=3 OR SR%=6 THEN F=1
3590 IF SR%>3 THEN  NUM=1:GOTO 3610
3600 NUM=0
3610 SS%=RND(0)*3+1
3611 IF M%<>4 AND M%<>8 THEN GOTO 3620
3614 REM SE IL MODO E' INFINITO ANNULLA LA PERSONA E CREA UNA FRASE OGGETTIVA
3619 ST$(I)="CREDO "+CM$(1,SG%,NUM,SS%):SR%=0:GOTO 3630
3620 ST$(I)=SO$(SG%,NUM,SS%):IF F<>1 THEN ST$(I)=""
3630 ST$(I)=ST$(I)+" "
3631 IFVG%(I)=3 AND NUM=0 THEN SR%=3
3632 IFVG%(I)=3 AND NUM=1 THEN SR%=6
3640 VR%(I)=SR%
3660 REM SE IL MODO E' PASSIVO SCEGLIE UN COMPLEMENTO D'AGENTE
3680 IF M>4 THEN  NP=3:GOTO 3740
3700 REM ALTRIMENTI UN COMPLEMENTO TRA GLI ALTRI CONSENTITI
3720 NP=CA%(M%,RND(0)*CA%(M,1)+2)
3730 IF NP=3 THEN GOTO 3720
3740 CC$(I)=CM$(NP,RND(0)*3+1,RND(0)+.5,RND(0)*3+1)
3760 REM SE E' L'ULTIMA FRASE C'E' IL SEGNO "."
3780 IF I=MAX% THEN CZ$(I)=CN$(5):GOTO 4020
3790 CD%=0
3810 REM DOPO UN IMPERATIVO FORZA LA SCELTA DI "UT"
3830 IF M%=3 OR M%=7 THEN CG%=3:GOTO 3930
3850 REM ALTRIMENTI SCEGLIE UNA CONGIUNZIONE CASUALE
3870 CG%=RND(0)*5+1
3880 IF CG%<>3 THEN GOTO 3970
3890 IF (M%=2 OR M%=6) THEN GOTO 3870
3910 REM SE ESCE UT FORZA LA SCELTA DI UN CONGIUNTIVO
3930 CD%=2:GOTO 3980
3950 REM SE ESCE IL PUNTO FORZA LA SCELTA DI UN IMPERATIVO
3970 IF CG%=5 THEN CZ$(I)=CN$(5):CD%=3:GOTO 4020
3980 CZ$(I)=" "+CN$(CG%)
4000 REM SCEGLIE UN PARADIGMA
4020 RP%=INT(RND(0)*10+1)
4030 B(I)=RP%
4040 PRE$=PD$(RP%,1)
4050 PER$=PD$(RP%,2)
4060 SUP$=PD$(RP%,3)
4070 INF$=PD$(RP%,4)
4090 REM E RICONOSCE LA CONIUGAZIONE DALL'INFINITO
4110 DI$=RIGHT$(INF$,3)
4120 IF DI$<>"ERE" THEN GOTO 4160
4130 IF RIGHT$(PRE$,2)="IO" THEN C=5:GOTO 4210
4140 IF RIGHT$(PRE$,2)="EO" THEN C=2:GOTO 4210
4150  C=3:GOTO 4210
4160 IF DI$="ARE" THEN C=1:GOTO 4210
4170 C=4
4190 REM PREPARA ALCUNE VARIABILI DAL PARADIGMA
4210 R1$=DES$(1,SR%):R2$=DES$(2,SR%)
4220 P5$=LEFT$(SUP$,LEN(SUP$)-2)+DGN$(SG%+NUM*3)
4230 I2$=LEFT$(INF$,LEN(INF$)-2)
4240 I3$=LEFT$(INF$,LEN(INF$)-3)
4250 DP$=LEFT$(PER$,LEN(PER$)-1)
4270 REM CONIUGAZIONE
4280 GOSUB 5570
4290 NEXT I
4300 RETURN
4320 REM SUBROUTINE DI STAMPA
```

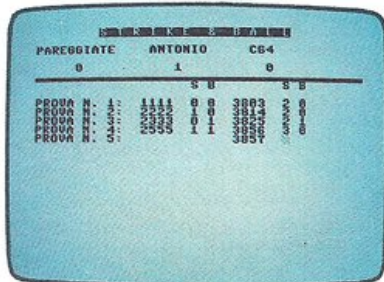
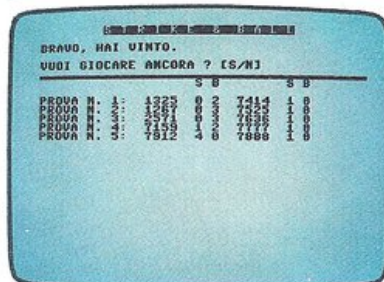
```
4340 W=0:BF$=""
4350 FOR I=1 TO MAX%
4360 BF$=BF$+ST$(I)
4380 REM COSTRUISCE LA MASCHERA DEL VERBO
4400 FOR K=1 TO LEN(SL$(I))
4410 BF$=BF$+"-"
4420 NEXT K
4430 BF$=BF$+" "+CC$(I)+CZ$(I)
4450 REM COMPOSIZIONE STAMPA
4470 GOSUB 6900
4480 NEXT I
4490 PRINT BF$
4500 RETURN
4520 REM SUBROUTINE FORMULAZIONE DOMANDE
4540 FOR I=1 TO MAX%
4550 ERR(I)=0
4560 POKE 214,13:PRINT:PRINT "                                        "
4570 PRINT CHR$(145);MODI$(VM%(I));" ";TM$(VT%(I));
4580 IF VR%(I)>0 THEN PRINT  VR%(I)  ;" PERS.";" ";SG$(VG%(I))       :GOTO4600
4590 PRINT"                "
4600 PRINT
4610 PRINT " DI ";CHR$(18);PD$(B(I),1);CHR$ (146);"     "
4630 REM RICERCA MASCHERA VIDEO
4670 IF PEEK(MV+W)=ASC("-") THEN GOTO 4690
4680 W=W+1:GOTO 4670
4690 K=0
4700 POKE MV+W+K,PEEK(MV+W+K)+128
4720 REM LETTURA TASTIERA
4740 GET TT$
4770 REM SIMULAZIONE BACKSPACE
4790 IF TT$<> CHR$(157) THEN GOTO 4880
4800 IF K<1 THEN GOTO 4740
4810 POKE MV+W+K,PEEK(MV+W+K)-128
4820 K=K-1
4830 RIS$=LEFT$(RIS$,LEN(RIS$)-1)
4840 GOTO 4700
4860 REM SIMULAZIONE CURSORE A DESTRA
4880 IF TT$<> CHR$(29) THEN GOTO 4940
4890 LL=PEEK(MV+W+K)-64
4895 IF LL=96 THEN LET TT$=" ":GOTO 4970
4900 TT$=CHR$(LL)
4920 REM CONTROLLO E VISUALIZZAZIONE CARATTERE
4940 IF TT$=CHR$(32) THEN  LL=96:GOTO 4970
4950 IF TT$<"A" OR TT$>"Z" THEN GOTO 4740
4960 LL=ASC(TT$)
4970 POKE MV+W+K,LL-64
4980 RIS$=RIS$+TT$
4990 TT$=""
5000 IF K<LEN(SL$(I)) -1 THEN K=K+1:GOTO 4700
5020 REM CONTROLLO RISPOSTA
5040 PRINT CHR$(19)
5050 POKE 214,19:PRINT
5060 IF RIS$<>SL$(I) THEN GOTO 5120
5070 PRINT "***     ESATTO     ***"
5080 FOR RIT=1 TO 1000:NEXT RIT
5090 PRINT CHR$(145);"                          "
5100 RIS$=""
5110 GOTO 5300
5120 PRINT "***     ERRATO     ***"
5130 FOR RIT=1 TO 1000:NEXT
5140 PRINT CHR$(145)"                         "
5150 RIS$=""
5170 REM CONTEGGIO ERRORI
5190 ERR(I)=ERR(I)+1
5200 IF ERR(I)<3 THEN GOTO 4690
5210 PRINT CHR$(145)"  TI ARRENDI? (S/N)?  ":PRINT CHR$(145);
5220 GET TT$
5230 IF TT$<>"S" THEN GOTO 5280
5240 PRINT " "SL$(I)"                  "
5250 FOR RIT=1 TO 1000:NEXT RIT
5260 PRINT CHR$(145);CHR$(145)"                          "
5270 GOTO 4690
5280 IF TT$="N" THEN PRINT "                              ":GOTO 4690
5290 GOTO 5220
5300 NEXT I
5310 RETURN
5330 REM SUBROUTINE COMMENTO
5350 PRINT CHR$(147)
5360 PRINT" PER CONTINUARE PREMI"
5370 PRINT"▓ LA BARRA SPAZIATRICE▓"
```

```
5380 PRINT
5390 FOR I=1 TO MAX%
5400 IF ERR(I)=0 THEN  GOTO 5510
5410 PRINT" NELLA FRASE ";I;" HAI COMMESSO"
5420 PRINT:PRINT ERR(I);" ERROR";
5430 IF ERR(I)=1 THEN PRINT"E.":GOTO 5450
5440 PRINT"I."
5450 PRINT
5460 PRINT" RIPASSA ";
5470 IF LEFT$(MODI$(VM%(I)),1)="I" THEN PRINT" L'";:GOTO 5490
5480 PRINT" IL ";
5490 PRINT MODI$(VM%(I))" "TM$(VT%(I))
5500 PRINT
5510 IF PEEK(197)<>60 THEN GOTO 5510
5520 NEXT I
5530 RETURN
5550 REM SUBROUTINE DI CONIUGAZIONE
5570 ON M% GOTO 5610,6120,6480,6610,5610,6120,6480,6610,6830
5590 REM INDICATIVO
5610 ON T% GOTO 5650,5770,5850,5940,6000,6060
5630 REM PRESENTE
5650 IF C=5 THEN R=6:GOTO 5670
5660  R=C+2
5670 A$=I3$+DES$(R,SR%)
5680 IF M%<5 THEN  SL$(I)=A$:RETURN
5690 IF SR% =1 THEN P$=A$+R2$:GOTO 5720
5700 IF SR% =2 AND(C=3 OR C=5) THEN P$=I2$+R2$:GOTO 5720
5710  P$=LEFT$(A$,LEN(A$)-LEN(R1$))+R2$
5720  SL$(I)=P$
5730 RETURN
5750 REM IMPERFETTO
5770 IF C=4 OR C=5 THEN A$=I3$+"IE":GOTO 5790
5780 A$=I2$
5790 IF M%<5 THEN SL$(I)=A$+"BA"+R1$:RETURN
5800  SL$(I)=A$+"BA"+R2$
5810 RETURN
5830 REM FUTURO
5850 IF C=3 THEN R=9:F$=I3$:GOTO 5890
5860 IF C<>4 AND C<>5 THEN GOTO 5880
5865 R=1
5875 IF SR%=1 THEN F$=I3$+"IA":GOTO 5890
5878 F$=I3$+"IE":GOTO 5890
5880 F$=I2$:R=7
5890 IF M%<5 THEN  SL$(I)=F$+DES$(R,SR%):RETURN
5900 SL$(I)=F$+DES$(R+1,SR%):RETURN
5920 REM PERFETTO
5940  R=11
5950 IF M%<5 THEN  SL$(I)=DP$+DES$(R,SR%):RETURN
5960  SL$(I)=P5$+" "+DES$(R+2,SR%):RETURN
5980 REM PIUCCHEPERFETTO
6000 R=11
6010 IF M%<5 THEN  SL$(I)=DP$+"ERA"+R1$:RETURN
6020 SL$(I)=P5$+" "+"ERA"+DES$(1,SR%):RETURN
6040 REM FUTURO ANTERIORE
6060  R=12:DP$=LEFT$(PER$,LEN(PER$)-1)
6070 IF M%<5 THEN  SL$(I)=DP$+DES$(R,SR%):RETURN
6080  SL$(I)=P5$+" "+DES$(R+2,SR%):RETURN
6100 REM CONGIUNTIVO
6120 ON T% GOTO 6160,6290,63999,6360,6430
6140 REM PRESENTE
6160 IF C=5 THEN R=6:GOTO 6210
6170  R=C+2
6180 IF C=1 THEN A$=I3$+"E"+R1$:GOTO 6220
6190 IF C=3 THEN A$= I3$+"A"+R1$:GOTO 6220
6200 IF C=2 OR C=4 THEN A$=I2$+"A"+R1$:GOTO 6220
6210 A$=I3$+"IA"+R1$
6220 P$=LEFT$(A$,LEN(A$)-LEN(R1$))+R2$
6230 IF M%<5 THEN SL$(I)=A$:RETURN
6240 SL$(I)=P$
6250 RETURN
6270 REM IMPERFETTO
6290 IF C=4 OR C=5 THEN A$=I3$+"IE":GOTO 6310
6300 A$=I2$
6310 IF M%<5 THEN  SL$(I)=INF$+R1$:RETURN
6320  SL$(I)=INF$+R2$:RETURN
6340 REM PERFETTO
6360 DP$=LEFT$(PER$,LEN(PER$)-1):R=11
6370 A$=DP$+"ERI"+R1$:P$=P5$+" SI"+R1$
6380 IF M%<5 THEN  SL$(I)=DP$+"ERI"+R1$:RETURN
6390  SL$(I)=P5$+" SI"+R1$:RETURN
6410 REM PIUCCHEPERFETTO
```

```
6430 IF M%<5 THEN  SL$(I)=PER$+"SSE"+R1$:RETURN
6440  SL$(I)=P5$+" ESSE"+R1$:RETURN
6460 REM IMPERATIVO
6480 IF SR%=5 THEN GOTO 6530
6490 IF C=1 THEN SL$(I)=I3$+"A":RETURN
6500 IF C=4 THEN LET SL$(I)=I3$+"I":RETURN
6510 SL$(I)=I3$+"E":RETURN
6520  SL$(I)=I3$+"I":RETURN
6530 IF C=1 THEN  SL$(I)=I3$+"A":GOTO 6560
6540 IF C=2 THEN  SL$(I)=I3$+"E":GOTO 6560
6550 SL$(I)=I3$+"I"
6560 SL$(I)=SL$(I)+"TE"
6570 RETURN
6590 REM INFINITO
6610 DS$=LEFT$(SUP$,LEN(SUP$)-2)
6615 ON T% GOTO 6650,63999,6720,6770
6630 REM PRESENTE
6650 IF M%<5 THEN  SL$(I)=INF$:RETURN
6660 IF C=3 OR C=5 THEN  SL$(I)=I3$+"I":RETURN
6670  SL$(I)=LEFT$(INF$,LEN(INF$)-1)+"I":RETURN
6690 REM FUTURO
6720 IF M%<5 THEN  SL$(I)=DS$+"UR"+DGN$(SG%+NUM*3+6)+" "+"ESSE":RETURN
6730 SL$(I)=DS$+"UM IRI":RETURN
6750 REM PERFETTO
6770 IF M%<5 THEN  SL$(I)=PER$+"SSE":RETURN
6780 SL$(I)=DS$+DGN$(SG%+NUM*3+6)+" ESSE":RETURN
6800 REM GERUNDIVO
6820 REM
6830 IF C=4 THEN A$=I2$+"END":GOTO 6850
6835 IF C=5 THEN A$=I3$+"IEND":GOTO 6850
6840 A$=I2$+"ND"
6850  SL$(I)=A$ +DGN$(SG%+NUM*3)+" "+DES$(13,SR%)
6860 RETURN
6880 REM SUBROUTINE DI COMPOSIZIONE
6900 BF$=BF$+" "
6910 IF LEN(BF$)<40 THEN RETURN
6920 SN=40
6930 IF MID$(BF$,SN,1)=CHR$(32)OR MID$(BF$,SN,1)=CHR$(160) THEN GOTO 6950
6940  SN=SN-1:GOTO 6930
6950 PRINT LEFT$(BF$,SN-1)
6960  BF$=RIGHT$(BF$,LEN(BF$)-SN)
6970 GOTO 6910
60960 REM SUBROUTINE:INIZIALIZZAZIONE COSTANTI
60980 REM CIRCUITO VIDEO
61000 VI=53248
61020 REM CIRCUITO SUONO
61040 SI=54272
61060 REM MEMORIA VIDEO
61080 MV=1024
61100 REM MEMORIA COLORE
61120 MC=55296
61180 REM INIZIALIZZAZIONE CHIP SUONO
61200 FOR I=0 TO 24
61210 POKE SI+I,0
61220 NEXT I
61230 RETURN
61980 REM SUBROUTINE:STAMPA IL TITOLO DEL PROGRAMMA MEMORIZZATO IN PG$
62000 GOSUB 61000
62010 POKE VI+32,15
62020 POKE VI+33,15
62030 PRINT "█████";
62040 PRINT TAB(6);"┌──────────────────────────────┐"
62050 FOR I=1 TO 5
62060 PRINT TAB(6);"│                              │"
62070 NEXT I
62080 PRINT TAB(6);"└──────────────────────────────┘"
62090 PRINT TAB(6);"████████DIGITA █RETURN▄ PER PROSEGUIRE"
62100 PRINT "█████"
62110 FOR I=1 TO 5
62120 PRINT TAB(7);
62130 FOR J=1 TO 26
62140 PRINT "██ ";
62150 NEXT J
62160 PRINT
62170 NEXT I
62190 REM ORA SCRIVO IL TITOLO
62210 PRINT "██████████";TAB((40-LEN(PG$))/2);"██";PG$
62220 GET Z9$
62230 IF Z9$<>CHR$(13) THEN GOTO 62220
62240 PRINT "██";
62250 RETURN
```

Strike and ball

Questo programma, che deriva dal famoso Mastermind, consiste in una gara fra due giocatori, nel corso della quale ciascuno di essi deve dapprima scegliere un numero segreto e poi cercare di indovinare il numero scelto dall'avversario. A turno ciascun giocatore dice un numero e l'avversario gli dà una risposta che comunica quanto il numero detto è vicino a quello segreto, specificando il numero delle cifre della risposta che sono presenti anche nel numero segreto ma non nella stessa posizione (cifre ball), e il numero delle cifre presenti anche nel numero segreto e che occupano la stessa posizione nei due numeri (cifre strike). Ad esempio, se il numero segreto è 4371 e il nostro avversario ha tentato di indovinarlo dicendo 2357, la nostra risposta sarà 1 cifra ball (il 7) e 1 cifra strike (il 3). Man mano che il gioco va avanti si può facilmente intuire come, in base alle risposte, alcune cifre devono essere eliminate dai tentativi successivi, altre devono essere solo cambiate di posizione (cifre ball), altre ancora, una volta individuate, non devono essere più spostate (cifre strike). Vince chi per primo indovina il numero segreto scelto dall'avversario.

Analisi del problema

Il programma che ora progetteremo dovrà consentire di giocare avendo come avversario il CBM-64. Il computer e l'utente sceglieranno ciascuno un numero a caso composto di quattro cifre comprese tra 0 e 9. A turno essi potranno poi effettuare dei tentativi, fino ad un massimo di quindici. Il calcolatore e l'utente comunicheranno di volta in volta all'avversario il numero di strike e di ball realizzati nell'ultimo tentativo.

La scelta del numero segreto da parte del CBM-64 avviene utilizzando la funzione RND(X), che genera numeri reali compresi tra 0 e 1 (estremi esclusi) uniformemente distribuiti nell'intervallo. Questa funzione permette di ottenere sequenze di numeri «effettivamente casuali», nel senso che una volta ottenuta una sequenza non è più possibile riottenerla da capo in modo identico, oppure sequenze «ripetibili» di numeri casuali, cioè sequenze i cui elementi possano essere riottenuti da capo, tutti e nello stesso ordine. Attribuendo alla funzione RND(X) argomento uguale a 0 si ottiene una sequenza di numeri ripetibili, che è quella che fa al caso nostro. Ogni numero generato è poi moltiplicato per 10, così da ottenere un numero casuale compreso tra 0 e 10 (estremi esclusi); infine si utilizza la funzione INT(X), che restituisce solo la parte intera del numero casuale generato. Così, con la sequenza di istruzioni

```
FOR I=0 TO 3 : NC(I)=INT (RND(0)*10) : NEXT I
```

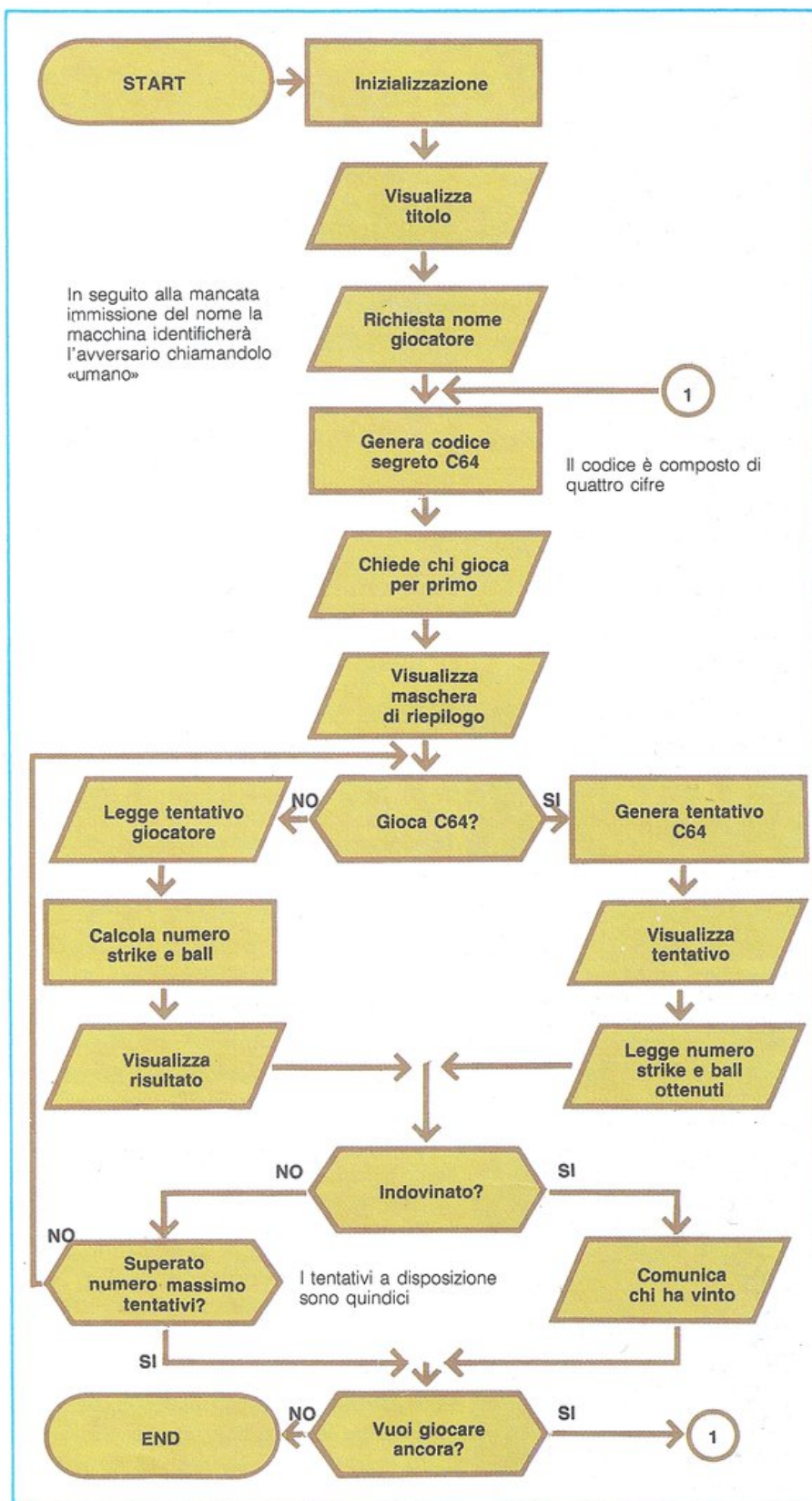
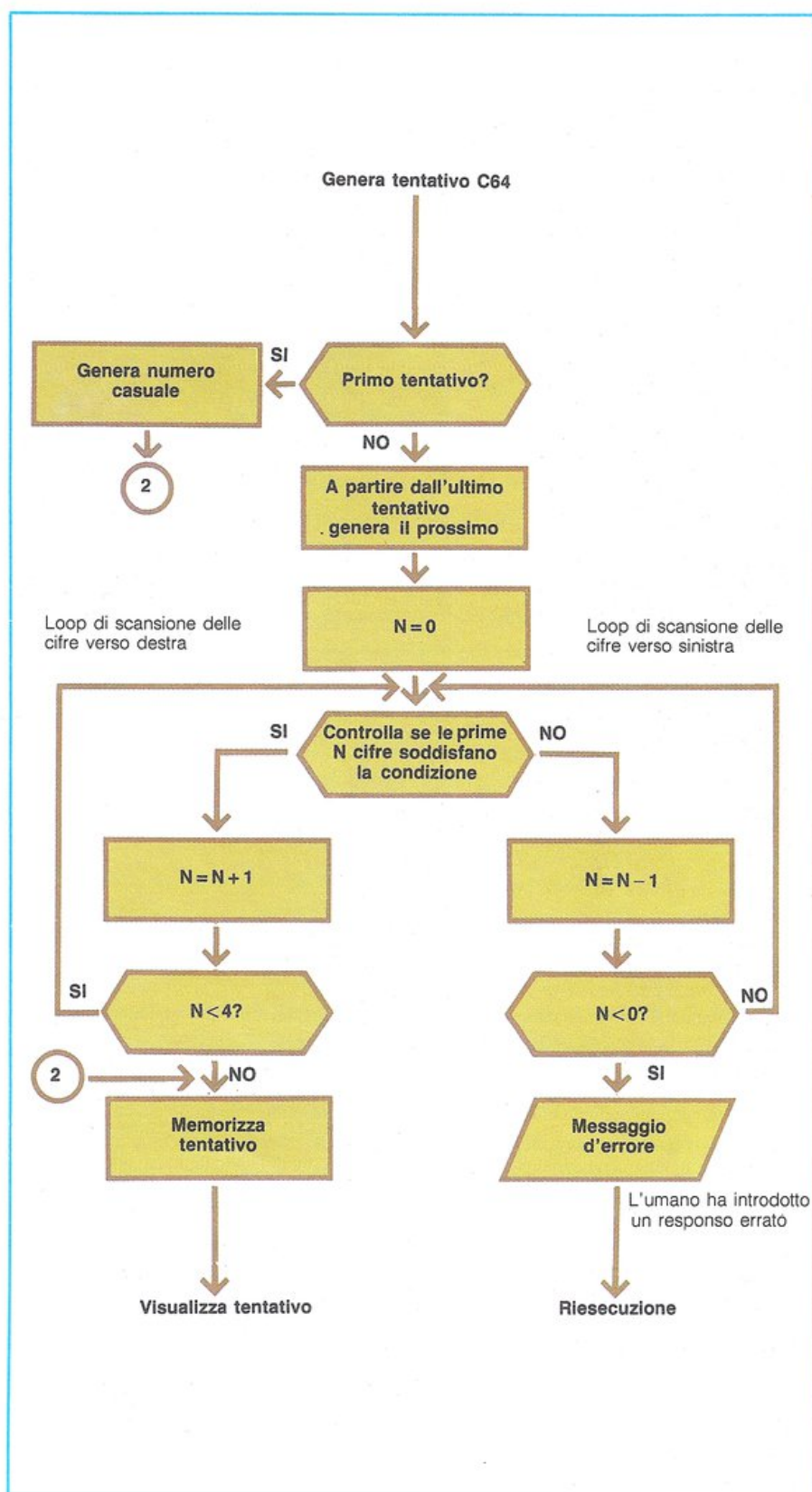
si otterrà la generazione del numero segreto da parte del CBM-64.

La maschera riepilogativa del gioco dovrà contenere, per ogni tentativo effettuato, il numero d'ordine del tentativo, la sequenza di cifre dichiarata in quel tentativo dal CBM-64 (matrice TC) o dal giocatore (matrice TU), il numero delle cifre strike e il numero delle cifre ball.

La determinazione del numero di strike e ball da parte del CBM-64 avverrà nel modo seguente: il programma andrà dapprima a verificare, posizione per posizione, se nel numero proposto e in quello segreto c'è la stessa cifra (determinazione delle cifre strike), quindi, dopo aver «marcato» le cifre già contate ponendole uguali a 0, passerà alla determinazione delle cifre ball, confrontando tutte le cifre del numero proposto con quelle del numero segreto; quando il programma avrà trovato due cifre uguali (e diverse da 0) le porrà uguali a 0 (per non ricontarle) e aumenterà di 1 il contatore delle cifre ball.

Al termine dei tentativi relativi ad un certo numero segreto (indovinato o no) il programma chiederà se si vuole tentare di indovinare un altro numero, e in caso affermativo rieseguirà l'intero procedimento.

Diagrammi di flusso

Dopo l'inizializzazione delle costanti e la visualizzazione del titolo, il programma chiede all'utente il nome e, dopo aver generato il proprio numero segreto, domanda chi deve giocare per primo. Visualizzata la maschera di riepilogo del gioco, se il primo a giocare è il C-64, il programma genera e poi visualizza il tentativo, restando in attesa della risposta, che l'utente deve digitare immettendo il numero di strike sotto la S e il numero di ball sotto la B. Una volta immessa, la risposta è memorizzata. Se a giocare per primo è invece l'«umano», la macchina si pone in attesa di leggere il tentativo. Effettuata la lettura, il programma calcola il numero di strike e ball realizzati e li visualizza nelle corrispondenti posizioni. Per decidere se il gioco è finito basta vedere se uno dei due giocatori ha realizzato con il suo tentativo quattro cifre strike. Comunque, prima di proclamare il vincitore, si deve verificare il numero dei tentativi effettuati dai due giocatori: se i due numeri sono uguali può essere proclamato il vincitore, altrimenti si deve concedere l'ultima possibilità al giocatore che ha effettuato un tentativo in meno. Nel caso che al termine ambedue i giocatori abbiano realizzato quattro strike la partita è dichiarata pari. La parte più interessante del programma è quella relativa

all'algoritmo illustrato qui a fianco, utilizzato dal computer per generare i suoi tentativi.
Prima di tutto la macchina controlla se è al suo primo tentativo; se SI, genera un numero composto di quattro cifre casuali attraverso la funzione RND e lo memorizza insieme al risultato ottenuto; se NO per generare il nuovo numero utilizza l'ultimo tra i tentativi precedenti, controllando se ognuna delle quattro cifre di tale numero soddisfa la seguente condizione: se la cifra è usata sempre nella stessa posizione all'interno di un nuovo tentativo, confrontando quest'ultimo con ciascuno dei tentativi precedenti non si devono avere numeri di strike e di ball superiori a quelli realizzati con il tentativo corrispondente. Se la condizione è soddisfatta, la cifra considerata può essere usata in quella posizione all'interno di un nuovo tentativo, altrimenti deve essere cambiata (incrementata di 1) e il test deve essere ripetuto. Se il numero di incrementi effettuati per una data cifra è pari a 10, deve essere incrementata la cifra che si trova a sinistra della posizione considerata. Se la cifra è quella più a sinistra, non esistendone altre prima di essa, il C-64 rileva la presenza di un errore nelle risposte fornite dall'umano e invia un messaggio tramite video, posizionandosi di nuovo all'inizio del programma per ricominciare il gioco. Dopo la verifica sulle quattro cifre, il tentativo è visualizzato e memorizzato insieme al risultato cui ha dato luogo.

Il programma

Il programma inizia alla linea 390 con il dimensionamento degli array per poi proseguire con la presentazione del titolo del gioco (linee 410, 420). In effetti la presentazione si ottiene chiamando la routine 3880 che prima di tutto inizializza le costanti, chiamando la routine 3610, e poi visualizza il titolo memorizzato in PG$.
Alla linea 460 sono inizializzate altre variabili di comodo, prima di passare (linee 470÷500) alla richiesta del nome del giocatore.
Alle linee 580÷600 il C-64 genera il proprio numero segreto, e poi visualizza la domanda «chi gioca per primo, tu o il C-64?» (linee 620÷660). Subito dopo (linee 690, 700) è visualizzata la maschera del gioco tramite la routine 2400.
Se a giocare per primo è il C-64 il programma salta (linea 720) all'istruzione 1230, altrimenti passa di seguito alla linea 750, in cui controlla se l'utente ha vinto.
In caso di risposta affermativa si va alla linea 1180, dove è prevista la chiamata alla routine 3040 che cancella le righe da 2 a 5 del video, dove il programma va a scrivere (linee 1190, 1200) la proclamazione del vincitore.
In caso di risposta negativa il programma continua alle linee 780÷810, dove, tramite la routine 2780, controlla se è stato superato il numero

```
100 REM ******************
110 REM **STRIKE & BALL**
120 REM ******************
140 REM            :VARIABILI UTILIZZATE
160 REM TU()       :MEMORIZZA TENTATIVI E RISULTATI DEL GIOCATORE
170 REM TC()       :MEMORIZZA TENTATIVI E RISULTATI DEL C64
180 REM TU         :NUMERO DI TENTATIVI DEL GIOCATORE
190 REM TC         :NUMERO DI TENTATIVI DEL C64
200 REM S          :NUMERO DI STRIKE
210 REM B          :NUMERO DI BALL
220 REM F1         :SE =1 ALLORA UNO DEI DUE GIOCATORI
221 REM            :HA INDOVINATO IL NUMERO DELL'ALTRO
230 REM F2         :SE =1 ALLORA IL TENTATIVO C64 E' UNA SOLUZIONE ACCETTABILE
240 REM N2()       :CONTIENE IL TENTATIVO DI C64 O DEL GIOCATORE
250 REM N1()       :CONTIENE IL NUMERO CON CUI CONFRONTARE N1()
260 REM N          :NUMERO DI CIFRE DI N2() DA CONFRONTARE CON N1()
270 REM NC()       :NUMERO SEGRETO C64
280 REM VU         :NUMERO DI PARTITE VINTE DAL GIOCATORE
290 REM VC         :NUMERO DI PARTITE VINTE DAL C64
300 REM PP         :NUMERO DI PARTITE PAREGGIATE
310 REM CC()       :INCREMENTI SUBITI DALLA CIFRA N-ESIMA DURANTE IL BACK-TRACKING
320 REM PG$        :NOME DEL PROGRAMMA
330 REM I,J,I1,J1:VARIABILI DI CICLO
340 REM NO$        :NOME DEL GIOCATORE
350 REM S$,K       :VARIABILI DI COMODO
360 REM R          :RIGA DI STAMPA
380 REM DIMENSIONO LE MATRICI
390 DIM TU(15,5),TC(15,5),N2(3),N1(3),NC(3),CC(3)
400 REM PRESENTAZIONE NOME PROGRAMMA
410 PG$="S T R I K E  &  B A L L"
420 GOSUB 3880
430 REM CARATTERI IN NERO
440 PRINT "⊐▮"
450 REM INIZIALIZZO IL NUMERO DI PARTITE VINTE DALL'UMANO, DAL C64 E PAREGGIATE
460 VU=0:VC=0:PP=0
470 PRINT "▮QUAL E' IL TUO NOME ? ";:ZL=17:GOSUB 3180
480 IF IN$="" THEN IN$="UMANO":PRINT IN$;
490 PRINT
500 NO$=IN$
510 FOR I=1 TO 500:NEXT I
520 REM ORA INIZIA IL GIOCO
530 REM FLAG INDOVINATO =0
540 F1=0:K=0
550 REM TENTATIVI UMANO E TENTATIVI C64 =0
560 TU=0:TC=0
570 REM GENERO IL NUMERO DEL C64
580 FOR I=0 TO 3
590 NC(I)=INT(RND(0)*10)
600 NEXT I
610 REM CHI E' IL PRIMO A GIOCARE ?
620 PRINT "⊐CHI GIOCA PER PRIMO,"
630 PRINT "▮▮TU O IL C64 ? ";:ZL=3:GOSUB 3180
640 IF IN$="" THEN IN$="C64":PRINT IN$;
650 PRINT
660 S$=IN$
670 FOR I=1 TO 500:NEXT I
680 REM VISUALIZZO LA MASCHERA DI GIOCO
690 PRINT "⊐"
700 GOSUB 2400
710 REM CONTROLLO CHI GIOCA PER PRIMO
720 IF S$="C64" THEN GOTO 1230
730 REM IL PRIMO A GIOCARE E' L'UMANO
740 REM SE IL FLAG F1 E' 1 ALLORA HA VINTO L'UMANO
750 IF F1 THEN GOTO 1180
760 REM LEGGO TENTATIVO DELL'UMANO
770 REM MI POSIZIONO SULLA RIGA OPPORTUNA
780 R=TU:GOSUB 2780
795 IF K=1 THEN GOTO 540
800 PRINT TAB(13);
810 ZL=4:GOSUB 3180
820 REM MEMORIZZO IL NUMERO LETTO E CALCOLO STRIKE E BALL
830 FOR I=0 TO 3
840 N2(I)=VAL(MID$(IN$,I+1,1))
850 TU(TU,I)=N2(I)
```

massimo di tentativi disponibili. Se ciò non è successo legge il tentativo dell'utente con la routine 3180, quindi memorizza il numero letto e calcola il numero di strike e di ball alle linee 830÷920, chiamando la routine 1680. Il risultato è visualizzato alla linea 940, chiamando la routine 2880. Dalla linea 990 alla 1200 c'è una serie di istruzioni necessarie al C-64 per sapere se l'utente ha vinto o se c'è stato un pareggio. Andando oltre troviamo la parte del programma dedicata ai tentativi del computer (linee 1230÷1630), che ripete ciò che si è fatto per l'utente con in più l'algoritmo di back-tracking (che genera il tentativo del C-64). Quest'ultimo è contenuto nella routine 1860, richiamata alla linea 1250. Terminato questo passo il programma salta alla linea 1080, dove è inserita l'istruzione che visualizza la domanda «vuoi giocare ancora?». Se la risposta è S il computer si posiziona all'inizio del programma, se la risposta è N il programma termina, ripristinando il colore del video e del cursore.

```
860 N1(I)=NC(I)
870 NEXT I
880 REM CALCOLO STRIKE E BALL
890 N=3:GOSUB 1680
900 REM MEMORIZZO STRIKE E BALL
910 TU(TU,4)=S
920 TU(TU,5)=B
930 REM VISUALIZZO RISULTATO
940 R=TU:GOSUB 2880
960 REM INCREMENTO TENTATIVI UMANO
970 TU=TU+1
980 REM SE STRIKE <> 4 TOCCA AL C64
990 IF S<>4 THEN GOTO 1230
1000 REM L'UMANO HA INDOVINATO, SE F1=1 ALLORA ANCHE IL C64
1010 REM AVEVA INDOVINATO E QUINDI E' UN PAREGGIO
1020 IF F1=0 THEN GOTO 1150
1030 REM E' UN PAREGGIO
1040 PP=PP+1
1050 REM CANCELLO DALLA LINEA 2 ALLA LINEA 5
1060 GOSUB 3040
1070 PRINT "███ABBIAMO PAREGGIATO."
1080 PRINT "█VUOI GIOCARE ANCORA ? [S/N] "
1090 GET S$:IF S$="" THEN GOTO 1090
1100 IF S$="S" THEN GOTO 540
1110 POKE 53280,14:POKE 53281,6: PRINT "██":END
1130 REM L'UMANO HA INDOVINATO. SE AL C64 MANCA UN TENTATIVO, LO FA
1140 REM IN CASO CONTRARIO HA VINTO L'UMANO
1150 IF TU>TC THEN F1=1:GOTO 1250
1160 REM L'UMANO HA VINTO
1170 REM CANCELLO LINEE 2..5
1180 GOSUB 3040
1190 PRINT "███BRAVO, HAI VINTO."
1200 VU=VU+1
1210 GOTO 1080
1220 REM TENTATIVO DEL C64.SE F1=1 HA VINTO IL C64
1230 IF F1 THEN GOTO 1610
1240 REM GENERO TENTATIVO C64
1250 GOSUB 1860
1255 IF K=1 GOTO 540
1260 REM SALVO IL TENTATIVO IN TC(.,.)
1270 FOR I=0 TO 3
1280 TC(TC,I)=N2(I)
1290 NEXT I
1300 REM VISUALIZZO TENTATIVO
1310 R=TC
1320 GOSUB 2780
1325 IF K=1 THEN GOTO 540
1330 GOSUB 2960
1340 REM LEGGO STRIKE E BALL
1350 POKE 214,8+R-1:PRINT
1360 PRINT TAB(30);
1370 ZL=1:GOSUB 3180
1380 IF IN$="" THEN GOTO 1350
1390 IF IN$<"0" OR IN$>"4" THEN PRINT "█ █";:GOTO 1370
1400 TC(TC,4)=VAL(IN$)
1410 S=VAL(IN$)
1420 PRINT SPC(1);
1430 GOSUB 3180
1440 IF IN$="" THEN GOTO 1350
1450 IF IN$<"0" OR IN$>"4" THEN PRINT "█ █";:GOTO 1430
1460 TC(TC,5)=VAL(IN$)
1470 B=VAL(IN$)
1480 IF S+B>4 THEN GOTO 1350
1490 REM INCREMENTO TENTATIVI C64
1500 TC=TC+1
1510 REM CONTROLLO SE SONO 4 STRIKE
1520 IF S<>4 THEN GOTO 750
1530 REM SONO 4 STRIKE.
1540 REM CONTROLLO SE ANCHE L'UMANO AVEVA INDOVINATO
1550 IF F1 THEN GOTO 1040
1560 REM C64 HA INDOVINATO. SE ALL'UMANO MANCA UN TENTATIVO, LO FA
1570 REM IN CASO CONTRARIO HA VINTO IL C64
1580 IF TC>TU THEN F1=1:GOTO 780
1590 REM HA VINTO C64.
```

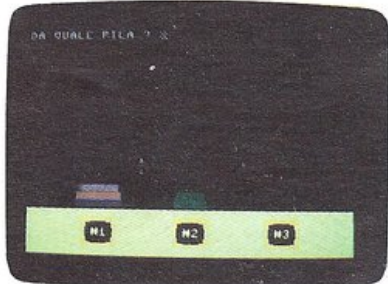
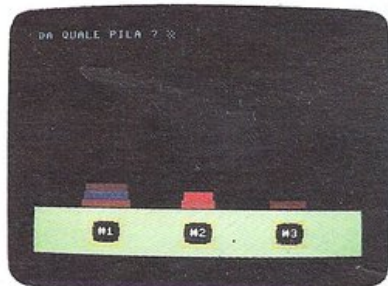
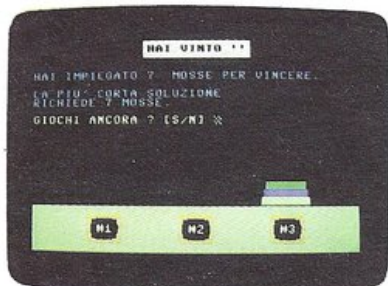
```
1600 REM CANCELLO LE LINEE 2..5
1610 GOSUB 3040
1620 PRINT "▒▒▒MI SPIACE, HAI PERSO."
1630 VC=VC+1
1631 PRINT"▒VUOI VEDERE IL MIO NUMERO? [S/N]"
1632 GETS$:IFS$="" THEN GOTO 1632
1633 IFS$="N"THEN PRINT"▒": GOTO 1650
1635 PRINT"▒":PRINT"IL MIO NUMERO ERA:    ";:FOR I=0 TO 3
1637 PRINT NC(I);"▒";:NEXT I:PRINT"▒▒"
1640 REM CHIEDO SE SI VUOL GIOCARE ANCORA
1650 GOTO 1080
1660 REM TERMINA IL PROGRAMMA PRINCIPALE. INIZIANO LE SUBROUTINES.
1670 REM ROUTINE:CALCOLA STRIKE E BALL
1680 S=0:B=0
1690 FOR I=0 TO 3:EL(I)=0:NEXT I
1700 FOR I=0 TO N
1710 IF N2(I)=N1(I) THEN S=S+1:EL(I)=1
1720 NEXT I
1730 REM HO CALCOLATO GLI STRIKE
1740 REM CALCOLO I BALL
1750 FOR I=0 TO N
1760 IF N2(I)=N1(I) THEN GOTO 1810
1770 J=0
1780 IF N2(I)=N1(J) THEN IF EL(J)=0 THEN B=B+1:EL(J)=1:GOTO 1810
1790 J=J+1
1800 IF J<4 THEN GOTO 1780
1810 NEXT I
1820 REM HO GLI STRIKE ED I BALL CHE LE N+1 CIFRE PIU' A SINISTRA DI N2(.)
1830 REM FANNO SULLE 4 CIFRE DI N1(.)
1840 RETURN
1850 REM ROUTINE:GENERA IL TENTATIVO DI C64
1860 IF TC>0 THEN GOTO 1930
1870 REM E' IL PRIMO TENTATIVO, GENERO IL NUMERO E LO ,METTO IN N2(.)
1880 FOR I=0 TO 3
1890 N2(I)=INT(RND(0)*10)
1900 NEXT I
1910 RETURN
1920 REM NON E' IL PRIMO TENTATIVO. PARTE IL BACK TRACKING
1930 FOR I=0 TO 3
1940 N2(I)=TC(TC-1,I)
1950 NEXT I
1960 REM AZZERO IL VETTORE CC(.)
1970 FOR I=0 TO 3
1980 CC(I)=0
1990 NEXT I
2000 REM ORA IN N2(.) C'E IL NUMERO DAL QUALE GENERARE LA PROSSIMA PROVA
2010 N=0
2020 REM CONTROLLO SE LE PRIME N CIFRE DI N2 SODDISFANO I RISULTATI PREC.
2030 GOSUB 2240
2040 IF F2=0 THEN GOTO 2100
2050 REM PROSEGUO QUESTA OPERAZIONE PER 4 VOLTE
2060 N=N+1:IF N<4 THEN GOTO 2030
2070 REM LA SOLUZIONE E' ACCETTABILE, TERMINA LA ROUTINE.
2080 RETURN
2090 REM IL TENTATIVO NON E' SOLUZIONE ACCETTABILE, CAMBIO CIFRA N-ESIMA
2100 N2(N)=N2(N)+1:IF N2(N)=10 THEN N2(N)=0
2110 CC(N)=CC(N)+1
2120 REM CONTROLLO SE LA CIFRA N-ESIMA E' STATA CAMBIATA 10 VOLTE
2130 IF CC(N)<10 THEN GOTO 2030
2140 REM E' STATA CAMBIATA 10 VOLTE
2150 CC(N)=0
2160 IF N>0 THEN N=N-1:GOTO 2100
2170 REM C'E' UN ERRORE NELLE RISPOSTE DELL'UMANO
2180 POKE 214,21:PRINT
2190 PRINT "▒ATTENZIONE, HAI COMMESSO UN ERRORE."
2200 PRINT "▒RICOMINCIAMO TUTTO DALL'INIZIO."
2210 FORI=1TO2000:NEXT I
2220 K=1: RETURN
2230 REM ROUTINE:CONTROLLA SE LA SOLUZIONE E' ACCETTABILE
2240 F2=0
2250 FOR I1=TC-1 TO 0 STEP -1
2260 REM COPIO TENTATIVO I-ESIMO IN N1(.)
2270 FOR J1=0 TO 3
2280 N1(J1)=TC(I1,J1):NEXT J1
```

```
2300 REM CALCOLO STRIKE E BALL
2310 GOSUB 1680
2320 REM CONTROLLO STRIKE E BALL
2330 IF S>TC(I1,4) THEN RETURN
2340 IF B>TC(I1,5) THEN RETURN
2350 REM LA SOLUZIONE SEMBRA ACCETTABILE
2360 NEXT I1:F2=1
2380 RETURN
2390 REM ROUTINE:VISUALIZZA LA MASCHERA DI GIOCO
2400 PRINT "⌧";
2410 REM SCRITTA STRIKE &  BALL
2420 PRINT TAB(8);"⌧S T R I K E  &  B A L L"
2430 REM PARTITE VINTE E PAREGGIATE
2440 PRINT "⌧";"PAREGGIATE";TAB(14);LEFT$(NO$,10);TAB(26);"C64"
2450 PRINT "⌧";TAB(4);PP;TAB(16);VU;TAB(27);VC
2460 REM LINEA DI SEPARAZIONE
2470 FOR I=1 TO 40
2480 PRINT "▬";:NEXT I
2500 REM SCRITTA "S B"
2510 PRINT "⌐";TAB(59);"S B";TAB(70);"S B"
2520 REM ORA VISUALIZZO I TENTATIVI FATTI ED I RISULTATI
2530 REM PRIMA SCRIVO LA PAROLA 'PROVA N.#:'
2540 R=1
2550 IF R>TU AND R>TC THEN GOTO 2620
2560 REM SCRIVO 'PROVA N.#:'
2570 GOSUB 2780
2580 R=R+1
2590 GOTO 2550
2600 REM ORA SCRIVO I RISULTATI DEI DUE GIOCATORI
2610 REM RISULTATI UMANO
2620 R=1
2630 IF R>TU THEN GOTO 2680
2640 GOSUB 2880
2650 R=R+1
2660 GOTO 2630
2670 REM RISULTATI COMMODORE
2680 R=1
2690 IF R>TC THEN GOTO 2740
2700 GOSUB 2960
2710 R=R+1
2720 GOTO 2690
2730 REM HO TERMINATO
2740 RETURN
2750 REM ROUTINE:VISUALIZZA LA SCRITTA 'PROVA N.#:' ALLA RIGA R-ESIMA
2760 REM MI POSIZIONO ALLA RIGA RELATIVA
2770 REM LA PRIMA RIGA E' LA 8
2780 IF TC<=15 THEN GOTO 2840
2790 REM IL NUMERO MASSIMO DI TENTATIVI E' STATO SUPERATO. STOP
2800 PRINT "⌐⌧IL NUMERO MASSIMO DI TENTATIVI E' STATO"
2810 PRINT "⌧SUPERATO. RICOMINCIAMO DALL' INIZIO"
2820 FOR I=1 TO 2000:NEXT I
2830 K=1:RETURN
2840 POKE 214,8+R-1:PRINT
2850 PRINT "PROVA N.";RIGHT$(STR$(R+1),2);":";
2860 RETURN
2870 REM ROUTINE:VISUALIZZA LA PROVA R-ESIMA DELL'UMANO ED IL RISULTATO
2880 POKE 214,8+R-1:PRINT
2890 PRINT TAB(13);
2900 FOR I=0 TO 3
2910 PRINT RIGHT$(STR$(TU(R,I)),1);
2920 NEXT I
2930 PRINT " ";STR$(TU(R,4));STR$(TU(R,5))
2940 RETURN
2950 REM ROUTINE:VISUALIZZA LA PROVA R-ESIMA DEL C64
2960 POKE 214,8+R-1:PRINT
2970 PRINT TAB(24);
2980 FOR I=0 TO 3
2990 PRINT RIGHT$(STR$(TC(R,I)),1);
3000 NEXT I
3020 RETURN
3030 REM ROUTINE: CANCELLO LE LINEE 2..5 DEL VIDEO
3035 REM 38 SPAZI
3040 PRINT "⌧⌧"
3050 PRINT "                                      "
```

```
3060 PRINT "                                        "
3070 PRINT "                                        "
3080 RETURN
3100 REM ROUTINE:GESTISCE L'INPUT DI UNA STRINGA ALFANUMERICA
3110 REM LE VARIABILI UTILIZZATE SONO:Z6,Z7,Z8,Z9,Z8$,IN$
3120 REM IN INGRESSO VUOLE IL VALORE ZL CHE E' LA LUNGHEZZA MASSIMA
3130 REM DELLA STRINGA DA LEGGERE
3140 REM IN USCITA DA' LA STRINGA LETTA IN IN$ E LA SUA LUNGHEZZA IN Z9
3160 REM CANCELLO LA ZONA DI SCHERMO IN CUI VERRA' DIGITATA LA STRINGA
3180 FOR Z8=1 TO ZL: PRINT " ";:NEXT Z8
3190 FOR Z8=1 TO ZL: PRINT "▌";:NEXT Z8
3200 IN$="":Z7=TI
3220 REM LEGGO UN CARATTERE
3240 GET Z8$:IF Z8$<>"" THEN GOTO 3340
3260 REM ACCENSIONE E SPEGNIMENTO DEL CURSORE
3280 IF Z7<TI AND NOT(Z6) THEN PRINT "▒▌";:Z6=NOT(Z6):Z7=TI+15
3290 IF Z7<TI AND Z6 THEN PRINT " ▌";:Z6=NOT(Z6):Z7=TI+15
3300 GOTO 3240
3320 REM E' STATO DIGITATO UN CARATTERE
3340 Z8=ASC(Z8$):Z9=LEN(IN$)
3360 REM SE NON E' UN CARATTERE ALFANUMERICO, DEVE ESSERE UN RETURN O DELETE
3380 IF NOT((Z8>47 AND Z8<58) OR (Z8>64 AND Z8<91)) THEN GOTO 3500
3400 REM CONTROLLO CHE NON SIA STATA SUPERATA LA LUNGHEZZA MASSIMA
3420 IF Z9=ZL THEN GOTO 3240
3440 REM LO AGGIUNGO ALLA STRINGA IN$
3460 IN$=IN$+Z8$:PRINT Z8$;:GOTO 3240
3480 REM SE E' UN RETURN, HO TERMINATO LA LETTURA
3500 IF Z8=13 THEN PRINT " ▌";:RETURN
3520 REM SE E' DELETE, CANCELLO IN IN$ E SUL VIDEO L' ULTIMO CARATTERE DIGITATO
3540 IF Z8=20 AND Z9>0 THEN IN$=LEFT$(IN$,Z9-1):PRINT " ▌▌";:GOTO 3240
3550 GOTO 3240
3570 REM ROUTINE:INIZIALIZZAZIONE COSTANTI
3590 REM CIRCUITO VIDEO
3610 VI=53248
3630 REM CIRCUITO SUONO
3650 SI=54272
3670 REM MEMORIA VIDEO
3690 MV=1024
3710 REM MEMORIA COLORE
3730 MC=55296
3750 REM COSTANTI DI USO COMUNE
3770 ZL=9
3790 REM INIZIALIZZAZIONE CHIP SUONO
3810 FOR I=0 TO 24
3820 POKE SI+I,0
3830 NEXT I
3840 RETURN
3860 REM ROUTINE:STAMPA IL TITOLO DEL PROGRAMMA MEMORIZZATO IN PG$
3880 GOSUB 3610
3890 POKE VI+32,15
3900 POKE VI+33,15
3910 PRINT "▒▒▒▒▒▒";
3920 PRINT TAB(6);"┌────────────────────────────┐"
3930 FOR I=1 TO 5
3940 PRINT TAB(6);"│                            │"
3950 NEXT I
3960 PRINT TAB(6);"└────────────────────────────┘"
3970 PRINT TAB(6);"▒▒▒▒▒▒▒▒DIGITA ▒RETURN▒ PER PROSEGUIRE"
3980 PRINT "▒▒▒▒▒▒"
3990 FOR I=1 TO 5
4000 PRINT TAB(7);
4010 FOR J=1 TO 26
4020 PRINT "▒▒ ";
4030 NEXT J
4040 PRINT
4050 NEXT I
4070 REM ORA SCRIVO IL TITOLO
4090 PRINT "▒▒▒▒▒▒▒▒▒";TAB((40-LEN(PG$))/2);"▒▒";PG$
4100 GET Z9$
4110 IF Z9$<>CHR$(13) THEN GOTO 4100
4120 PRINT "▒▒";
4130 RETURN
```

La torre di Hanoi

In un monastero del Tibet — così racconta un'antica leggenda orientale — si trovano tre aste conficcate nel pavimento di una stanza: sulla prima asta all'inizio del mondo erano disposti 64 dischi concentrici, in modo che i più piccoli stessero al di sopra dei più grandi. Dall'inizio del mondo i monaci spostano un disco alla volta da un'asta all'altra con l'intento di portarli tutti su un'altra delle due aste, ma facendo sì che un disco più grande non sia mai posto su uno più piccolo. Secondo la leggenda quando i monaci termineranno il loro compito il mondo finirà.
Per quanto ci riguarda, limiteremo l'altezza delle nostre torri ad un massimo di 8 dischi. Il nostro programma dovrà essere in grado di rappresentare sul video la pila iniziale di dischi e il movimento di un singolo disco da una pila all'altra, di accettare ed eseguire le nostre mosse, verificandone l'esattezza, e di comunicare quando siamo riusciti a spostare l'intera pila di dischi, risolvendo così il problema.

Analisi del problema

La parte più complessa del programma sarà sicuramente quella che si occuperà della rappresentazione dei dischi, che saranno definiti come sprites (da cui il massimo di 8 dischi); gli sprites consentiranno di simulare in modo semplice lo spostamento di un disco da una pila all'altra.
Dopo l'inizializzazione delle variabili e la stampa del titolo, il programma dovrà creare gli sprites necessari a rappresentare il numero di dischi voluto. A tal fine, dovrà chiedere all'utente con quanti dischi vuole giocare, accettando un numero compreso tra 2 e 8. Per semplicità, stabiliamo di creare comunque sempre tutti e 8 gli sprites, fissandone la forma, le dimensioni, il colore e la posizione iniziale sul video. Non tutti gli sprites saranno però attivati, cioè visibili: saranno attivati solo gli N dischi più grandi, cioè quelli che si trovano più in basso nella pila (N è il numero di dischi con cui l'utente vorrà giocare).
Ricordiamo che per fissare le caratteristiche degli sprites si può operare nel modo seguente:

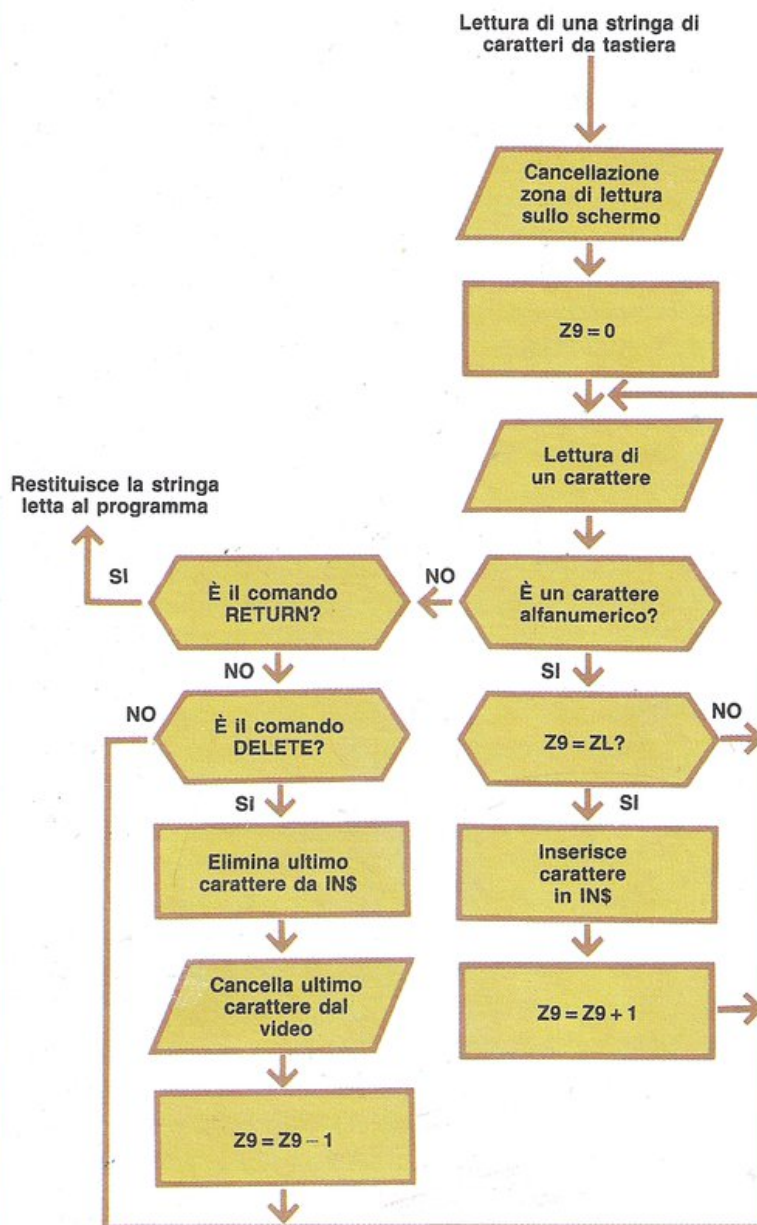
- la forma di ogni sprite è contenuta in un blocco di memoria di 64 bytes consecutivi; l'indirizzo di partenza del blocco deve essere inserito in una delle locazioni 2040÷2047
- lo sprite può essere espanso in orizzontale o in verticale ponendo pari ad 1 il bit corrispondente ad esso nei registri 53277 o 53271
- il codice del colore deve essere inserito in uno dei registri 53287÷53294
- le coordinate degli sprites sul video debbono essere inserite nelle coppie di registri poste dalla locazione 53248 alla 53263.

Dopo aver definito gli 8 sprites e attivato quelli necessari, il programma chiederà all'utente da quale pila a quale pila vuole spostare un disco; dopo aver verificato se la mossa è corretta, dovrà rappresentare il movimento del disco sul video, operando sui valori contenuti nella coppia di registri di posizionamento dello sprite corrispondente. Dovrà inoltre tenere conto del numero delle mosse effettuate, e controllare se il rompicapo è stato risolto, ovvero se l'intera pila è stata spostata sulla seconda o sulla terza asta. Se ciò è avvenuto, chiederà se si vuole effettuare una nuova partita, altrimenti terminerà l'esecuzione.
Il programma dovrà offrire infine sia la possibilità di terminare il gioco in un qualunque istante (ad esempio premendo il tasto F come risposta alla richiesta di una nuova mossa), sia la comunicazione, al termine di una partita, del numero di mosse effettuate e del numero minimo di mosse necessarie, dato dalla formula

2 ↑Numero dei dischi − 1.

Diagrammi di flusso

Il diagramma presentato è abbastanza semplice nella sua struttura, ed è più utile descrivere in dettaglio una sua parte che sarà utilizzata anche nell'ambito di altri programmi considerati in questo volume.

Il blocco in questione, riportato qui a fianco, è quello relativo alla richiesta di una mossa, che è stato realizzato come blocco di lettura di una sequenza di caratteri lunga al massimo ZL in una variabile di tipo stringa IN$; al termine della lettura la variabile Z9 contiene il numero dei caratteri letti. La routine accetta un carattere alla volta. Se il carattere è alfabetico o numerico, è visualizzato ed inserito in fondo alla stringa; in caso contrario, se si tratta di un RETURN la lettura della stringa cessa, se è un DELETE è cancellato da IN$ e dal video l'ultimo carattere letto e si torna (come nel caso in cui il carattere non è né RETURN né DELETE) a leggere un nuovo carattere. In IN$ sono inseriti un massimo di ZL caratteri; eventuali caratteri successivi sono visualizzati sul monitor ma non inseriti nella stringa.

Il programma

Le linee 200÷290 controllano se il programma, secondo quanto indicato nell'avvertenza riportata a pag. 37, è stato caricato a partire dalla locazione di memoria 16384. Se non è così è visualizzato un messaggio di avvertimento. Se tutto è a posto il programma ha inizio con la linea 570 che insieme alla 580 provvede, chiamando la routine 4270 ed in sequenza la 4000, ad inizializzare le costanti e a stampare il titolo.
Alle linee 620÷700 sono inizializzate le variabili, sono dimensionati gli array ed è impostato il colore del video. Subito dopo (linea 740), tramite la routine 2950, sono inizializzati gli sprites. Guardando le istruzioni presenti nella routine 2950 possiamo osservare che per la maggior parte sono costituite da POKE che, facendo riferimento ad una locazione base VI=53248, inseriscono determinati valori all'interno dei registri degli sprites.
Alle linee 780÷920 il programma chiede all'utente il numero dei dischi con cui vuole giocare.
Le linee comprese dalla 960 alla 1120 inizializzano in modo casuale i colori degli otto dischi che sono poi posizionati (linee 1160÷1240) sulla prima asta.
Proseguendo nel programma, si disegna prima il piano su cui poggeranno i dischi (linee

```
100 REM ****************************
110 REM *****LA TORRE DI HANOI*****
120 REM ****************************
140 REM ATTENZIONE: PRIMA DI MANDARE IN ESECUZIONE QUESTO PROGRAMMA OCCORRE
150 REM IMPOSTARE I SEGUENTI COMANDI:
160 REM POKE 44,64
170 REM POKE 16384,0
180 REM CHE SERVONO A SPOSTARE LA BASE DEL PROGRAMMA BASIC ALLA LOCAZIONE 16384
190 REM CONTROLLO CHE QUANTO SCRITTO SIA STATO ESEGUITO
200 IF PEEK(44)=64 THEN GOTO 570
210 REM QUANTO DETTO NON E' AVVENUTO
220 PRINT "■■■■"
230 PRINT "ATTENZIONE, HAI DIMENTICATO"
240 PRINT "■DI IMMETTERE I SEGUENTI COMANDI:"
250 PRINT "■■■1) POKE 44,64"
260 PRINT "■■2) POKE 16384,0"
270 PRINT "■■PRIMA DI IMMETTERLI RICORDA"
280 PRINT "■DI SALVARE IL PROGRAMMA, SE QUESTO"
290 PRINT "■NON E' GIA' PRESENTE SU NASTRO,"
300 PRINT "■E POI DI RICARICARLO": END
310 REM VARIABILI UTILIZZATE
320 REM PG$      :NOME DEL PROGRAMMA
330 REM I,J,I1   :CONTATORI DI CICLO
340 REM SI       :BYTE DI SINISTRA DI UNA RIGA CHE COMPONE UNA FIGURA VELOCE
350 REM CE       :BYTE CENTRALE DI UNA RIGA CHE COMPONE UNA FIGURA VELOCE
360 REM DE       :BYTE DI DESTRA DI UNA RIGA CHE COMPONE UNA FIGURA VELOCE
370 REM PO       :LOCAZIONE NELLA QUALE E' MEMORIZZATA LA FIGURA VELOCE
380 REM N        :NUMERO DI DISCHI CON I QUALI SI GIOCA
390 REM T        :PILA DI PARTENZA PER UNO SPOSTAMENTO
400 REM F        :PILA DI ARRIVO PER UNO SPOSTAMENTO
410 REM DN$      :SE STAMPATO, POSIZIONA IL CURSORE ALLA 21-MA RIGA
420 REM MO       :NUMERO DI MOSSE FATTE
430 REM TM       :VARIABILE DI COMODO
440 REM PR$      :MEMORIZZA LE DOMANDE CHE VENGONO FATTE AL GIOCATORE
450 REM DS       :NUMERO DEL DISCO DA SPOSTARE
460 REM SX,SY    :REGISTRI SPOSTAMENTO FIGURE VELOCI
470 REM PP       :ASCISSA DI PARTENZA PER UN DISCO
480 REM PA       :ASCISSA DI ARRIVO PER UN DISCO
490 REM SP       :INDICA SE LO SPOSTAMENTO E' VERSO DESTRA O VERSO SINISTRA
500 REM CL()     :CODICI DI NEWE PER I DISCHI
510 REM AL()     :OTTO POSSIBILI ORDINATE DI UN DISCO
520 REM PO()     :ASCISSE DELLE TRE PILE
530 REM P()      :NUMERO DI DISCHI E IDENTIFICATORI DEI DISCHI DI OGNI PILA
550 REM VISUALIZZO IL NOME DEL PROGRAMMA ED INIZIALIZZO LE COSTANTI COMMODORE
570 PG$="LA TORRE DI HANOI"
580 GOSUB 4270
600 REM INIZIALIZZO LA LUNGHEZZA MASSIMA DELLE STRINGHE IN INGRESSO
620 ZL=1
640 REM DIMENSIONO VETTORI E MATRICI
660 DIM CL(8),AL(8),PO(3),P(3,8)
680 REM SFONDO NERO
700 POKE VI+32,0:POKE VI+33,0
720 REM INIZIALIZZO LE FIGURE VELOCI
740 GOSUB 2950
760 REM INIZIA IL PROGRAMMA
780 PRINT "■";
790 PRINT:PRINT "■QUANTI DISCHI (DA 2 A 8) ? ";
810 REM LEGGO UNA STRINGA ALFANUMERICA LUNGA ZL DALLA TASTIERA
830 GOSUB 3570:PRINT
840 IF IN$="" THEN PRINT "NE USERAI 3":IN$="3"
860 REM RITARDO
880 FOR I=1 TO 500:NEXT I
890 IF IN$>="0" AND IN$<="9" THEN 910
900 PRINT "NON USARE ";IN$;" PER FAVORE.":GOTO 790
910 N=VAL(IN$):IF N>8 THEN PRINT"NON POSSONO ESSERE PIU' DI 8.":GOTO 790
920 IF N<2 THEN PRINT"NON ESSERE RIDICOLO!":GOTO 790
940 REM METTO NEL VETTORE CL(8) OTTO NUMERI DIVERSI COMPRESI TRA 1 E 15
960 FOR I=1 TO 8
970 CL(I)=0
980 NEXT I
990 FOR I=1 TO 8
1000 T=INT(RND(0)*15+1)
1010 J=1
1020 IF CL(J)=T THEN 1000
1030 J=J+1
1040 IF J<I THEN 1020
1050 CL(I)=T
1060 NEXT I
1080 REM ORA INIZIALIZZO I COLORI DELLE FIGURE VELOCI
1100 FOR I=1 TO 8
1110 POKE VI+38+I,CL(I)
1120 NEXT I
1140 REM STAMPANDO DN$, POSIZIONO IL CURSORE ALL'INIZIO DELLA 21-MA RIGA
1160 DN$="■■■■■■■■■■■■■■■■■■■■■"
1180 REM AZZERO LA MATRICE P(3,8)
1200 FOR I=1 TO 3:FOR J=0 TO 8:P(I,J)=0:NEXT J,I
1220 REM GLI N DISCHI SONO TUTTI SULLA PRIMA PILA
1240 P(1,0)=N
1260 REM DISEGNO IL PIANO SU CUI POGGERANNO I DISCHI
1280 PRINT"■";DN$;"■■"
```

1280÷1330), quindi si attivano gli sprites rappresentanti i dischi con cui si vuole giocare. L'attivazione degli sprites è fatta alle linee 1380÷1440. La linea 1580, chiamando la routine 2510, cancella le quattro righe più in alto del video, dove andranno visualizzate le domande poste al giocatore e le mosse che si vogliono effettuare (linee 1620÷1740). Le linee 1790, 1800 calcolano la posizione del disco da spostare mentre le linee 1840÷2010 effettuano lo spostamento. Come si può vedere lo spostamento è diviso in tre blocchi: nel primo è realizzato lo spostamento verso l'alto (1840÷1860), nel secondo quello orizzontale (1900÷1950) e nel terzo quello verso il basso (1990÷2010). Tutti gli spostamenti utilizzano istruzioni POKE agenti sui registri che controllano la posizione orizzontale (POKE SX, I) e quella verticale (POKE SY, I). Le linee 2050÷2110 memorizzano la mossa appena fatta. Per finire, le linee 2150÷2280 controllano se il gioco è terminato, visualizzano alcune informazioni, come per esempio il numero di mosse, e controllano se si vuole giocare ancora. In caso affermativo il programma salta alla linea 620, altrimenti, tramite la routine 4540, disattiva gli sprites e ripristina il colore del video.

```
1290 FOR I=1 TO 5:PRINT
1300 PRINT"▒                                        ▮▮▮ ";:NEXT I
1310 PRINT DN$;"▒▮▮▮▮▮▮▮▮▒▒ ▒▒▒ ▒▒▮▮▮▮▮ ▒#1▒ ▒▮▮▮▮▮▒ ▒▒▒ ▒▒▒";
1320 PRINT  "▮▮▮▮▮▮▮▮▒▒ ▒▒▒ ▒▒▮▮▮▮▮ ▒#2▒ ▒▮▮▮▮▮▒ ▒▒▒ ▒▒▒";
1330 PRINT  "▮▮▮▮▮▮▮▮▒▒ ▒▒▒ ▒▒▮▮▮▮▮ ▒#3▒ ▒▮▮▮▮▮▒ ▒▒▒ ▒▒";
1350 REM FACCIO APPARIRE GLI N DISCHI RICHIESTI CHE SONO STATI POSIZIONATI
1360 REM DALLA ROUTINE DI INIZIALIZZAZIONE FIGURE VELOCI
1380 FOR I=1 TO N
1390 POKE VI+21,2↑(I-1) OR PEEK(VI+21)
1410 REM LA PILA 1, AL POSTO I-ESIMO, HA IL DISCO I-ESIMO
1430 P(1,I)=I
1440 NEXT I
1460 REM METTO A ZERO IL NUMERO DI MOSSE FATTE
1480 MO=0
1500 REM INIZIA IL CICLO DI GIOCO
1510 REM RITARDO DI UN SECONDO
1530 TM=TI+60
1540 IF TI<TM THEN 1540
1560 REM CANCELLO LE PRIME QUATTRO LINEE DEL VIDEO
1580 GOSUB 2510
1600 REM RICHIEDO LA PILA DI ORIGINE E QUELLA DI DESTINAZIONE
1620 PRINT"▒"
1630 PR$="▒DA QUALE PILA ? "
1650 REM LEGGO UN NUMERO DA UNO A TRE
1670 GOSUB 2330:IF IN$="" THEN GOTO 1580
1680 F=VAL(IN$)
1690 IF P(F,0)=0 THEN PRINT "▒QUESTA PILA E' VUOTA.":GOTO 1530
1700 PR$="A QUALE PILA ? ":GOSUB 2330:IF IN$="" THEN 1580
1710 T=VAL(IN$)
1720 IF F=T THEN PRINT "▒NON E' POSSIBILE.":GOTO 1530
1730 IF P(T,0)=0 THEN 1790
1740 IF P(F,P(F,0))<P(T,P(T,0)) THEN PRINT "▒MOSSA NON VALIDA.":GOTO 1530
1760 REM SPOSTO UN DISCO DALLA PILA F ALLA PILA T
1770 REM CALCOLO QUALE DISCO DEVO SPOSTARE
1790 DS=P(F,P(F,0))-1
1800 SX=VI+2*DS:SY=SX+1
1820 REM SPOSTO IL DISCO IN VERTICALE VERSO L'ALTO
1840 FOR I=AL(P(F,0)-1) TO 100 STEP -1
1850 POKE SY,I
1860 NEXT I
1880 REM ORA LO SPOSTO IN ORIZZONTALE
1900 PP=PO(F):PA=PO(T)
1910 SP=1
1920 IF F>T THEN SP=-1
1930 FOR I=PP TO PA STEP SP
1940 POKE SX,I
1950 NEXT I
1970 REM ORA LO ABBASSO FINO AL PRIMO POSTO LIBERO
1990 FOR I=100 TO AL(P(T,0))
2000 POKE SY,I
2010 NEXT I
2030 REM MEMORIZZO LO SPOSTAMENTO
2050 P(T,0)=P(T,0)+1
2060 P(T,P(T,0))=P(F,P(F,0))
2070 P(F,0)=P(F,0)-1
2090 REM INCREMENTO IL NUMERO DI MOSSE FATTE
2110 MO=MO+1
2130 REM CONTROLLO SE E' STATA FORMATA UNA PILA DI N DISCHI AI POSTI 2 E 3
2150 IF P(2,0)<>N AND P(3,0)<>N THEN 1530
2160 GOSUB 2510
2170 PRINT "▒▒                    ▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬ "
2180 PRINT "                    ▒HAI VINTO !!▒"
2190 PRINT "                    "▒▬▬▬▬▬▬▬▬▬▬▬▬▬▬▒ "
2200 PRINT "▒▒HAI IMPIEGATO";MO;" MOSSE";" PER VINCERE."
2210 T=2↑N-1:PRINT "▒LA PIU' CORTA SOLUZIONE "
2220 PRINT "RICHIEDE";T;"MOSSE."
2230 PRINT "▒▒GIOCHI ANCORA ? [S/N] ";:GOSUB 3570:PRINT "▒"
2240 IF IN$="N" THEN GOSUB 4540:END
2260 REM RIPARTO CON IL PROGRAMMA
2280 RESTORE:CLR:GOSUB 4000:POKE VI+21,0:GOTO 620
2300 REM ROUTINE:LEGGE UN NUMERO DALLA TASTIERA E CONTROLLA CHE SIA COMPRESO
2310 REM TRA UNO E TRE. SE VIENE DIGITATO 'F' IL PROGRAMMA TERMINA
2330 PRINT PR$;
2340 GOSUB 3570:PRINT:IF IN$="" THEN RETURN
2350 IF IN$="F" THEN GOSUB 4540:END
2360 IF IN$>="1" AND IN$<="3" THEN RETURN
2370 PRINT "QUESTA PILA NON C'E'."
2390 REM RITARDO
2410 FOR I=1 TO 500:NEXT I
2430 REM FACCIO APPARIRE DI NUOVO LA DOMANDA
2450 PRINT "▒                                   "
2460 PRINT "▒▒";PR$;" ";
2470 GOTO 2340
2490 REM ROUTINE:CANCELLA LE PRIME QUATTRO LINEE DEL VIDEO
2510 PRINT "▒";
2520 FOR I=1 TO 4:PRINT "▪                                        ":NEXT I
2530 PRINT "▒";
2540 RETURN
2560 REM ROUTINE:MEMORIZZA UN DISCO CON DIMENSIONI DATE DA SI,CE,DE
2580 FOR I1=0 TO 20
```

```
2590 POKE PO+I1,0
2600 NEXT I1
2610 FOR I1=21 TO 39 STEP 3
2620 POKE PO+I1,SI
2630 POKE PO+1+I1,CE
2640 POKE PO+2+I1,DE
2650 NEXT I1
2660 FOR I1=42 TO 62
2670 POKE PO+I1,0
2680 NEXT I1
2690 RETURN
2710 REM DIMENSIONI E LOCAZIONI DI PARTENZA DEI DISCHI
2730 DATA 1,255,128,2048
2740 DATA 3,255,192,2112
2750 DATA 7,255,224,2176
2760 DATA 15,255,240,2240
2770 DATA 31,255,248,2304
2780 DATA 63,255,252,2368
2790 DATA 127,255,254,2432
2800 DATA 255,255,255,2496
2820 REM POSIZIONI VERTICALI DEI DISCHI DI UNA PILA
2840 DATA 146,153,160,167,174,181,188,195
2860 REM ROUTINE:GESTIONE DI 8 FIGURE VELOCI
2870 REM MEMORIZZATE DA 2048 A 2559
2890 REM CARICO IN 2040 -> 2047
2900 REM GLI INDIRIZZI DEI BLOCCHI
2910 REM DA 64 BYTES NEI QUALI MEMORIZZO
2920 REM LE 8 FIGURE VELOCI
2930 REM MESSAGGIO PER L'UTENTE
2950 PRINT "▒ATTENDI UN ATTIMO":PRINT "▒PER FAVORE"
2955 PRINT"▒▒SE VUOI ARRENDERTI PRIMA DI TERMINARE":PRINT"▒IL GIOCO PREMI 'F'"
2960 FOR I=0 TO 7
2970 POKE 2040+I,39-I
2980 NEXT I
3000 REM MEMORIZZO LE OTTO FIGURE
3010 REM DALLA PIU' GRANDE ALLA PIU' PICCOLA
3030 FOR I=0 TO 7
3040 READ SI,CE,DE,PO
3050 GOSUB 2580
3060 NEXT I
3080 REM INIZIALIZZAZIONE CIRCUITO
3090 REM VIC-II PER LA GESTIONE DI
3100 REM OTTO FIGURE VELOCI
3120 REM NESSUNO ABILITATO
3140 POKE VI+21,0
3160 REM BIT ALTI DELLE POSIZIONI ORIZZONTALI A ZERO
3180 POKE VI+16,0
3200 REM TUTTI ESPANSI IN ORIZZONTALE
3220 POKE VI+23,0
3230 POKE VI+29,255
3250 REM PREDISPONGO TUTTE LE POSIZIONI VERTICALI
3270 FOR I=7 TO 0 STEP -1
3280 READ AL(I)
3290 NEXT I
3300 FOR I=0 TO 7
3310 POKE VI+2*I+1,AL(I)
3320 NEXT I
3340 REM CARICO IL VETTORE POSIZIONI ORIZZONTALI
3360 PO(1)=72:PO(2)=160:PO(3)=247
3380 REM ORA POSIZIONO TUTTA LA PILA
3390 REM DI 8 ANELLI SULLA PILA 1
3410 FOR I=0 TO 7
3420 POKE VI+I*2,72
3430 NEXT I
3450 REM HO TUTTO PRONTO PER ESSERE VISUALIZZATO
3470 RETURN
3490 REM ROUTINE:GESTISCE L'INPUT DI UNA STRINGA ALFANUMERICA
3500 REM LE VARIABILI UTILIZZATE SONO:Z6,Z7,Z8,Z9,Z8$,IN$
3510 REM IN INGRESSO VUOLE IL VALORE ZL CHE E' LA LUNGHEZZA MASSIMA DELLA
3520 REM STRINGA DA LEGGERE
3530 REM IN USCITA DA' LA STRINGA LETTA IN IN$ E LA SUA LUNGHEZZA IN Z9
3550 REM CANCELLO LA ZONA DI SCHERMO IN CUI VERRA' DIGITATA LA STRINGA
3570 FOR Z8=1 TO ZL: PRINT " ";:NEXT Z8
3580 FOR Z8=1 TO ZL: PRINT "▒";:NEXT Z8
3590 IN$="":Z7=TI
3610 REM LEGGO UN CARATTERE
3630 GET Z8$:IF Z8$<>"" THEN 3730
3650 REM ACCENZIONE E SPEGNIMENTO DEL CURSORE
3670 IF Z7<TI AND NOT(Z6) THEN PRINT "▒▒";:Z6=NOT(Z6):Z7=TI+15
3680 IF Z7<TI AND Z6 THEN PRINT " ▒";:Z6=NOT(Z6):Z7=TI+15
3690 GOTO 3630
3710 REM E' STATO DIGITATO UN CARATTERE
3730 Z8=ASC(Z8$):Z9=LEN(IN$)
3750 REM SE NON E' UN CARATTERE ALFANUMERICO, DEVE ESSERE UN RETURN O CMD
3770 IF NOT((Z8>47 AND Z8<58) OR (Z8>64 AND Z8<91)) THEN GOTO 3890
3790 REM CONTROLLO CHE NON SIA STATA SUPERATA LA LUNGHEZZA MASSIMA
3810 IF Z9=ZL THEN GOTO 3630
3830 REM LO AGGIUNGO ALLA STRINGA IN$
3850 IN$=IN$+Z8$:PRINT Z8$;:GOTO 3630
3870 REM SE E' UN RETURN, HO TERMINATO LA LETTURA
```

```
3890 IF Z8=13 THEN PRINT " ▮";:RETURN
3910 REM SE E' DELETE, CANCELLO IN IN$ E SUL VIDEO L' ULTIMO CARATTERE DIGITATO
3930 IF Z8=20 AND Z9>0 THEN IN$=LEFT$(IN$,Z9-1):PRINT " ▮▮";:GOTO 3630
3940 GOTO 3630
3960 REM RE:INIZIALIZZAZIONE COSTANTI
3980 REM CIRCUITO VIDEO
4000 VI=53248
4040 SI=54272
4060 REM MEMORIA VIDEO
4080 MV=1024
4100 REM MEMORIA COLORE
4120 MC=55296
4140 REM COSTANTI DI USO COMUNE
4160 ZL=9
4180 REM INIZIALIZZAZIONE CHIP SUONO
4200 FOR I=0 TO 24
4210 POKE SI+I,0
4220 NEXT I
4230 RETURN
4250 REM ROUTINE:STAMPA IL TITOLO DEL PROGRAMMA MEMORIZZATO IN PG$
4270 GOSUB 4000
4280 POKE VI+32,15
4290 POKE VI+33,15
4300 PRINT "▮▮▮▮▮";
4310 PRINT TAB(6);"┌──────────────────────────┐"
4320 FOR I=1 TO 5
4330 PRINT TAB(6);"│                          │"
4340 NEXT I
4350 PRINT TAB(6);"└──────────────────────────┘"
4360 PRINT TAB(6);"▮▮▮▮▮▮▮DIGITA ▮RETURN▮ PER PROSEGUIRE"
4370 PRINT "▮▮▮▮▮"
4380 FOR I=1 TO 5
4390 PRINT TAB(7);
4400 FOR J=1 TO 26
4410 PRINT "▮▮ ";
4420 NEXT J
4430 PRINT
4440 NEXT I
4460 REM ORA SCRIVO IL TITOLO
4480 PRINT "▮▮▮▮▮▮▮▮▮";TAB((40-LEN(PG$))/2);"▮▮";PG$
4490 GET Z9$
4500 IF Z9$<>CHR$(13) THEN GOTO 4490
4510 PRINT "▮▮",
4520 RETURN
4530 REM ROUTINE: FINE PROGRAMMA
4540 PRINT"▮": POKE VI+21,0
4550 POKE VI+21,0
4560 POKE 53280,14
4570 POKE 53281,6
4580 PRINT "▮"
4590 PRINT "▮"
4600 RETURN
```

Grafica monodimensionale

Questo programma è in grado di visualizzare l'andamento del grafico di una funzione di una variabile la cui espressione analitica sia specificata all'interno del programma stesso.
La funzione da rappresentare dovrà essere inserita nel programma a cura dell'utente, in una zona appositamente prevista, altrimenti sarà utilizzata, per default, la funzione SINX/X.

Analisi del problema

Una funzione è un'entità matematica che fa corrispondere ad ogni valore di una certa variabile X il valore di una seconda variabile Y. Per indicare che una certa variabile dipende da un'altra si può scrivere l'espressione Y=f(X); X è chiamata variabile indipendente e Y variabile dipendente.
L'eguaglianza esprime l'esistenza di un legame di natura qualsiasi fra la variabile Y e la variabile X, in base al quale ad ogni valore assegnato alla X, appartenente ad un certo insieme di numeri I, viene a corrispondere un valore ed uno solo per la Y.
Per ottenere la rappresentazione grafica di una certa funzione Y=f(X) su un piano, prima di ogni cosa bisogna fissare un intervallo di definizione per la X, ad esempio quello contenuto fra i valori a e b (si indica così: [a,b]). Attribuendo alla X un valore qualunque X_1 compreso in [a,b], e chiamando $Y_1=f(X_1)$ il corrispondente valore di Y, i due numeri X_1,Y_1 rappresentano le coordinate di un punto P del piano. L'insieme di tutti i punti che si ottengono facendo variare la X nell'intervallo [a,b], cioè l'insieme di tutti i punti del piano le cui coordinate soddisfano l'equazione Y=f(X), è il grafico o diagramma della funzione f(X).
Particolare attenzione bisogna prestare al fatto che non tutte le funzioni forniscono punti rappresentabili sul piano per qualsiasi valore della variabile indipendente. Un esempio è dato dalla funzione Y=1/X, che non è definita per X=0. Per questo motivo si dovrà evitare di chiedere al computer di rappresentare le funzioni nei punti che non appartengono al loro insieme di definizione.
Dopo aver creato le condizioni per la visualizzazione, il programma chiederà all'utente se vuole rivedere ed eventualmente modificare l'espressione analitica della funzione:

- in caso affermativo, visualizzerà la linea del programma che la contiene
- in caso negativo, richiederà i limiti dell'intervallo sull'asse X in cui si vuole conoscere il diagramma (valore minimo e massimo di X).

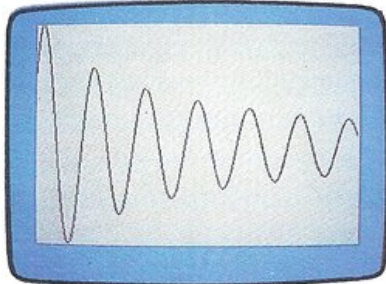
Dopo l'immissione dei due dati sarà calcolato il passo di variazione della X per il calcolo dei valori di Y; seguirà poi il calcolo dei valori massimo e minimo assunti dalla funzione (cioè dalla Y) nell'intervallo dell'asse X specificato. Resterà così individuata un'area rettangolare contenente il grafico della funzione in questione. Il fatto che sia l'utente a fissare l'intervallo sull'asse X in cui rappresentare la funzione consentirà di espandere o di ridurre a piacere l'intervallo dei valori considerati, in modo da poter avere la rappresentazione dell'andamento della funzione in ogni punto in cui è definita e con il grado di dettaglio desiderato. Il grafico della funzione sarà rappresentato in modo dettagliato facendo uso della gestione del video in bit-map, caratteristica del CBM-64. In questo modo di funzionamento i singoli punti del video sono indirizzabili dal programma, che è in grado di gestire l'immagine punto per punto, a differenza di quanto avviene nel modo normale, nel quale il calcolatore può inviare al video soltanto caratteri.

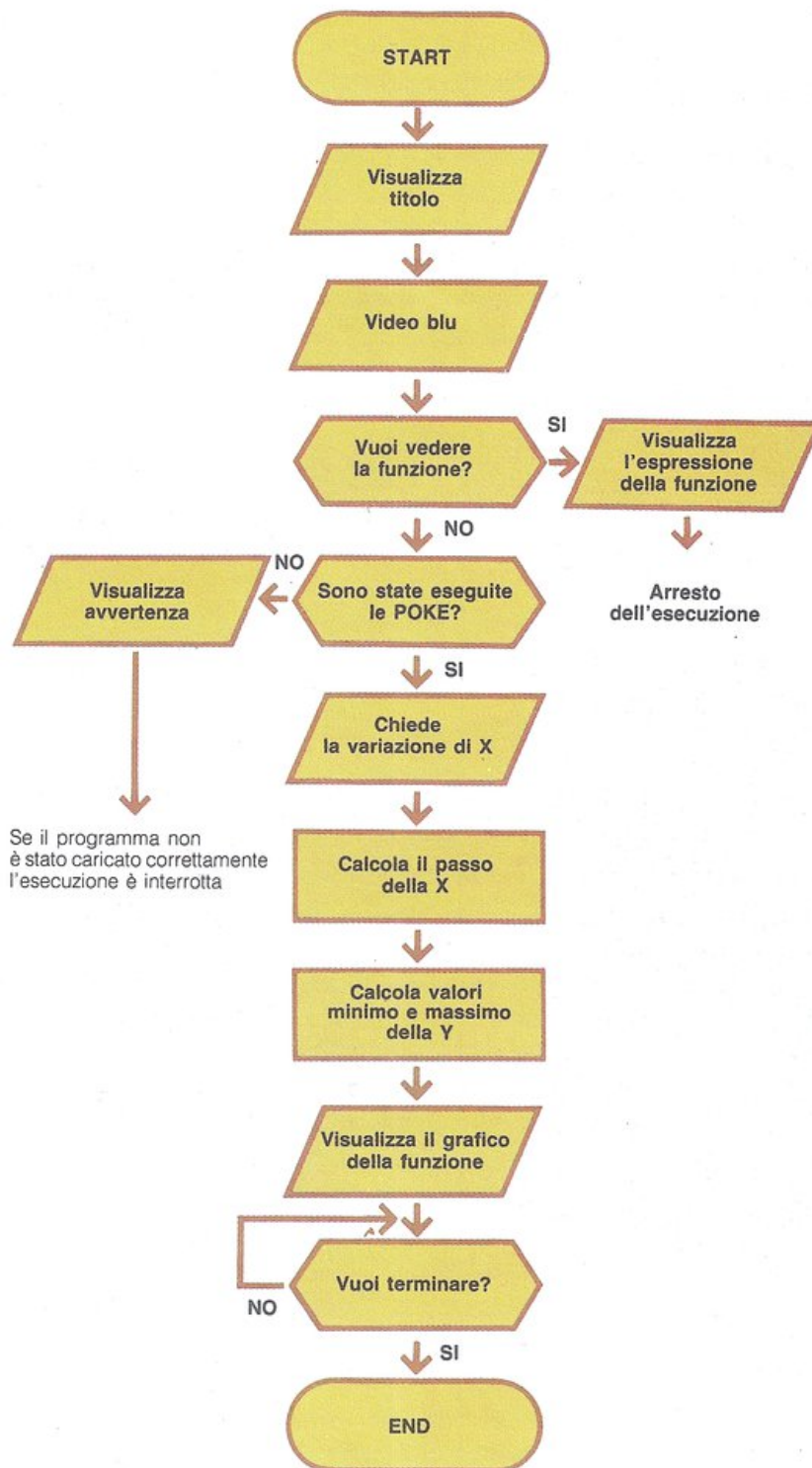
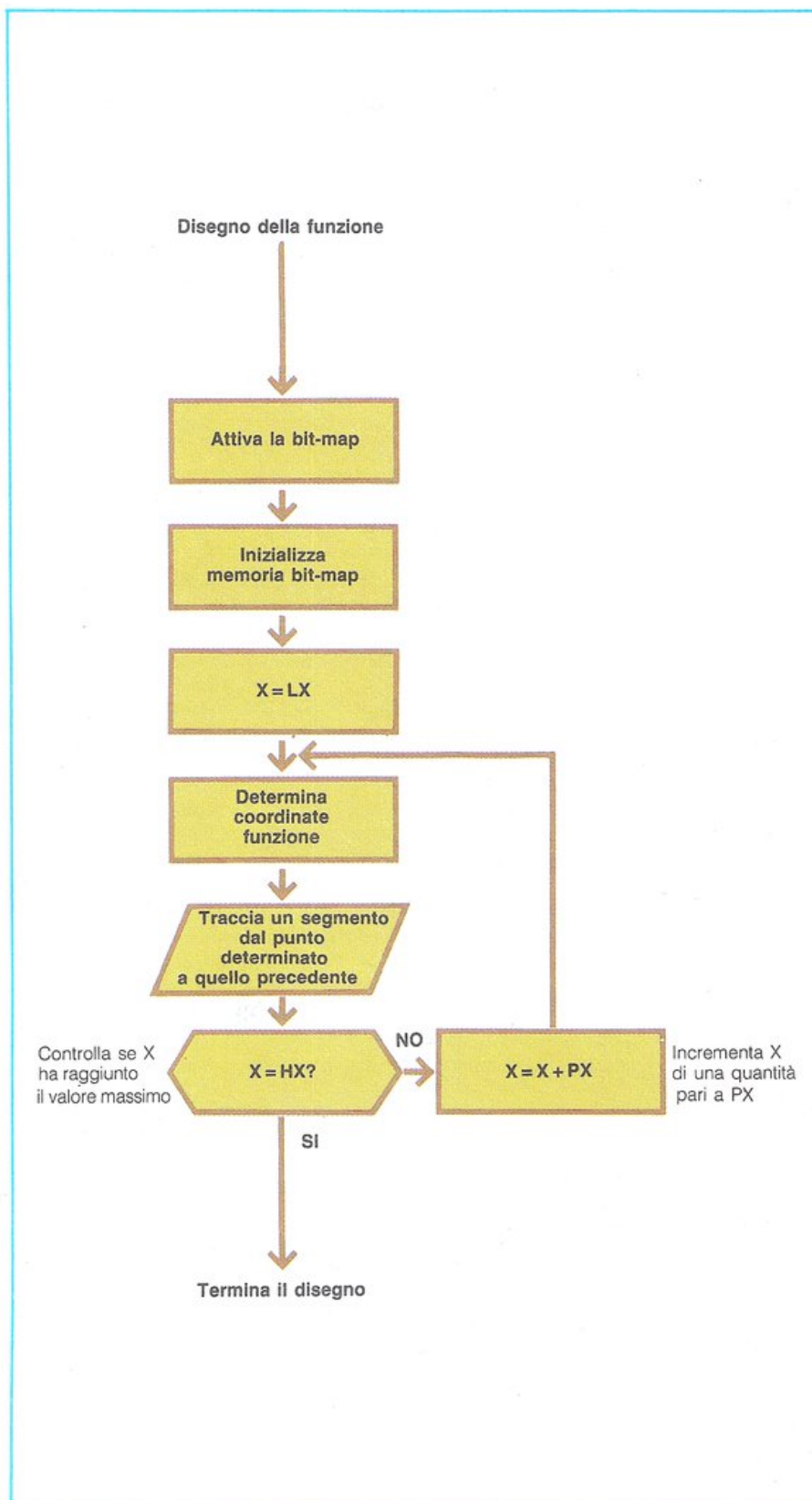
ATTENZIONE, LA FUNZIONE DA VISUALIZZARE
E' MEMORIZZATA ALLA LINEA 340.

DIGITA 'C' PER VISUALIZZARLA
O PER GIOCARE ANCORA

DIGITA 'S' PER MODIFICARLA

DIGITA 'F' PER TERMINARE

[C/S/F] ?

Diagrammi di flusso

Dopo aver visualizzato il titolo, il programma cambia il colore dello sfondo e chiede all'utente se vuole vedere o modificare l'espressione analitica della funzione; se SI, la linea del programma che contiene la definizione è visualizzata tramite il comando LIST. Poiché è stato utilizzato questo comando, il programma si interrompe, e per mandarlo di nuovo in esecuzione è necessario ridare il RUN. Se la risposta è NO, il programma verifica se l'utente ha effettuato le operazioni POKE preliminari al caricamento del programma, necessarie per allocarlo correttamente in memoria, al fine di riservare una parte della memoria alla gestione in bit-map del video. Se le POKE sono state eseguite, il programma chiede all'utente di definire l'intervallo della variabile X sul quale si vuole tracciare il grafico della funzione. L'ampiezza di questo intervallo determina la scelta del passo di valutazione del valore di Y, cioè la differenza tra due valori successivi della X in cui è calcolata la funzione. Sono quindi calcolati i valori massimo e minimo assunti dalla funzione nell'intervallo considerato, al fine di determinare la scala della variabile Y sullo schermo.

Il grafico della funzione è visualizzato secondo l'algoritmo rappresentato nel diagramma di flusso di questa pagina. Prima di tutto è attivata la bit-map ed è inizializzata la parte di memoria utilizzata. Poi, partendo dal minimo valore di X nell'intervallo di valutazione, è determinato il valore di f(X); si passa quindi a determinare il valore f(X+PX), dove PX è il passo calcolato per la variabile X. Infine i due punti f(X) e f(X+PX), relativi a due valutazioni successive, sono collegati tracciando un segmento.

Il programma

La prima linea del programma (340) definisce la funzione di cui si vuole tracciare il grafico; segue, alla linea 370, la chiamata alla subroutine che inizializza gli indirizzi e le costanti, e che visualizza il titolo del programma. Queste due routines si trovano in coda al programma, a partire dalle linee 2000 e 2270. Le due istruzioni POKE contenute nella linea 390 impostano per lo sfondo il colore blu. Alle linee 410÷455 troviamo un gruppo di istruzioni PRINT che visualizzano alcune informazioni e la richiesta se l'utente vuole vedere ed eventualmente modificare la funzione da graficare: se l'operatore digita C si va all'istruzione 620 dove inizia il programma vero e proprio, se invece digita S il programma va alla linea 476 e lista il contenuto della linea 340, cioè l'espressione della funzione. Con l'esecuzione dell'istruzione LIST 340 (linea 479) il programma è sospeso. Per rimandarlo in esecuzione è necessario immettere di nuovo il comando RUN. Alla linea 500 si verifica se sono state immesse le istruzioni POKE necessarie prima di caricare il programma. Se ciò non è avvenuto, il computer avvisa l'utente che bisogna rimemorizzare il programma su disco o su nastro (linee 570, 580), immettere i comandi POKE 44,64 e POKE 16384,0 (linee 540, 550) e quindi ricaricare il programma

```
100 REM ************************
110 REM GRAFICA MONODIMENSIONALE
120 REM ************************
140 REM VARIABILI UTILIZZATE
160 REM PG$   :NOME PROGRAMMA
170 REM I,J   :CONTATORI DI CICLO
180 REM LX    :VALORE MINIMO ASSUNTO DALLA VARIABILE X
190 REM HX    :VALORE MASSIMO ASSUNTO DALLA VARIABILE X
200 REM PX    :PASSO DELLA X
210 REM LY    :VALORE MINIMO ASSUNTO DALLA Y
220 REM HY    :VALORE MASSIMO ASSUNTO DALLA Y
230 REM FX,FY:FATTORI DI SCALA PER LA VISUALIZZAZIONE
240 REM X,Y   :CONTATORI DI CICLO
250 REM T     :VARIABILE DI COMODO
270 REM ATTENZIONE, QUESTO PROGRAMMA INIZIA ALLA LOCAZIONE 16384.
280 REM QUINDI, PRIMA DI CARICARLO, OCCORRE IMPOSTARE LE SEGUENTI OPERAZIONI:
290 REM POKE 44,64
300 REM POKE 16384,0.
310 REM ORA IL PROGRAMMA VIENE ESEGUITO CORRETTAMENTE
320 REM LA FUNZIONE MONODIMENSIONALE
330 REM MEMORIZZATA ALLA LINEA 340 E' QUELLA DA VISUALIZZARE
340 DEF FNY(X)=SIN(X)/X
350 REM INIZIA IL PROGRAMMA
360 PG$="GRAFICA MONODIMENSIONALE"
370 GOSUB 2270
380 REM SFONDO BLU
390 POKE VI+32,6:POKE VI+33,6
400 REM STAMPO INFORMAZIONI
410 PRINT "■";
420 PRINT "ATTENZIONE, LA FUNZIONE DA VISUALIZZARE"
430 PRINT "■E' MEMORIZZATA ALLA LINEA 340."
440 PRINT "■■■DIGITA 'C' PER VISUALIZZARLA"
441 PRINT "■D PER GIOCARE ANCORA"
450 PRINT "■■■DIGITA 'S' PER MODIFICARLA"
455 PRINT "■■■DIGITA 'F' PER TERMINARE"
460 ZL=1:PRINT "■■[C/S/F] ? ";:GOSUB 1570
470 IF IN$="F" THEN GOTO 870
475 IF IN$<>"S" THEN GOTO 500
476 PRINT "■■■"
477 PRINT "DOPO AVER MODIFICATO LA FUNZIONE"
478 PRINT "■PER MANDARE IN ESECUZIONE IL PROGRAMMA"
479 PRINT "■DEVI DIGITARE DI NUOVO ■ RUN ■■■■■":LIST 340
480 REM RICORDIAMO CHE L'OPERAZIONE 'LIST', SOSPENDE L'ESECUZIONE DEL PROGRAMMA
490 REM CONTROLLO CHE IL PROGRAMMA SIA SCRITTO DALLA LOCAZIONE 16384 IN POI
500 IF PEEK(44)=64 THEN GOTO 620
510 PRINT "■■■"
520 PRINT "ATTENZIONE, HAI DIMENTICATO"
530 PRINT "■DI IMMETTERE I SEGUENTI COMANDI:"
540 PRINT "■■■1) POKE 44,64"
550 PRINT "■■2) POKE 16384,0
560 PRINT "■■PRIMA DI IMMETTERLI RICORDA"
570 PRINT "■DI SALVARE IL PROGRAMMA, SE QUESTO"
580 PRINT "■NON E' GIA' PRESENTE SU NASTRO,"
590 PRINT "■E POI DI RICARICARLO"
600 END
610 REM ORA INIZIA IL PROGRAMMA
620 INPUT "■■■■■VARIAZIONE DELLA X [MIN,MAX]";LX,HX
630 IF LX>=HX THEN GOTO 620
640 REM CALCOLO IL PASSO DELLA X
650 PX=(HX-LX)/319
660 REM CALCOLO IL MINIMO ED IL MASSIMO PER Y
670 LY=FNY(LX):HY=LY
680 FOR X=LX TO HX STEP PX*4:T=FNY(X)
690 IFT<LYTHENLY=T:NEXT X:GOTO 720
700 IFT>HYTHENHY=T:NEXT X:GOTO 720
710 NEXT X
720 FX=319/(HX-LX)
730 REM FATTORE DI SCALA PER Y
740 FY=199/(HY-LY)
750 REM DISEGNO LA FUNZIONE
```

(linea 590). Come si è detto, i comandi specificati servono per memorizzare il programma in una zona di memoria che consenta la contemporanea gestione della bit-map. Alla linea 620 inizia il programma vero e proprio. L'utente fissa i valori minimo e massimo della X, e il programma (linea 650) calcola il passo PX. Alle linee 670÷710 si calcolano i valori massimo e minimo per la funzione Y. In particolare, l'istruzione 710 interviene se i due NEXT inseriti nelle linee 690, 700 dopo le istruzioni IF non chiudono il ciclo iniziato alla linea 680. Le istruzioni 720 e 740 fissano il fattore di scala per la X e per la Y, mentre le istruzioni 760÷880 visualizzano il grafico della funzione. In questo ultimo gruppo di istruzioni sono richiamate diverse routines. La prima è quella che attiva la bit-map (linee 950÷1020); ad essa segue la routine che cancella la memoria bit-map (linee 1040, 1050). Lo scopo della cancellazione è quello di avere ogni punto dello schermo libero. La routine della linea 1070 fissa il colore dello sfondo grigio chiaro (codice 15) e quella della linea 1150 traccia un segmento che unisce gli ultimi due punti calcolati [quello precedente, di coordinate (XP,YP), e quello seguente, di coordinate (XS,YS)]. Infine, la routine 1450 disattiva la gestione del video in modalità bit-map.

```
760 GOSUB 950
770 GOSUB 1040
780 CO=15:GOSUB 1070
790 XP=0:YP=(FNY(LX)-LY)*FY
800 FOR X=LX TO HX STEP PX
810 XS=(X-LX)*FX:YS=(FNY(X)-LY)*FY
820 GOSUB 1150
830 XP=XS:YP=YS
840 NEXT X
850 GET IN$:IF IN$="" THEN GOTO 850
860 GOSUB 1450
865 IF IN$="C" THEN GOTO 390
870 PRINT "♥"
880 POKE VI+32,14:END
900 REM I SOTTOPROGRAMMI PER LA GESTIONE DELLA GRAFICA, SONO SCRITTI
910 REM IN MODO DA RENDERE IL PIU' VELOCE POSSIBILE LA LORO AZIONE
930 REM ROUTINE:INIZIALIZZA IL MODO VIDEO BIT-MAP E LE VARIABILI INTERESSATE
940 REM ATTIVO IL MODO BIT-MAP
950 POKE VI+17,PEEK(VI+17) OR 32
960 REM LA MEMORIA BIT-MAP E' ALLA LOCAZIONE 8192
970 POKE VI+24,PEEK(VI+24) OR 8
980 BM=8192:Z0=8:Z1=320:Z2=7:Z3=1/Z0
990 FOR I=0 TO 7
1000 Z3(I)=2↑I
1010 NEXT I
1020 RETURN
1030 REM ROUTINE:CANCELLO LA MEMORIA BIT-MAP
1040 Z9=0:FORI=BMTOBM+7999:POKEI,Z9:NEXTI
1050 RETURN
1060 REM ROUTINE:IMPOSTO LO SFONDO NEL COLORE CO
1070 FOR I=1024TO2023:POKEI,CO:NEXT I
1080 RETURN
1090 REM ROUTINE:ACCENDE IL PUNTO DI COORDINATE XC,YC
1100 BY=BM+INT(YC*Z3)*Z1+INT(XC*Z3)*Z0+(YCANDZ2)
1110 POKEBY,PEEK(BY)ORZ3(Z2-(XCANDZ2))
1120 RETURN
1130 REM ROUTINE:TRACCIA UNA RETTA DA XP,YP A XS,YS
1140 REM LA RETTA E' PARALLELA ALL'ASSE XC?
1150 IF YP=YS THEN GOTO 1280
1160 REM LA RETTA E' PARALLELA ALL'ASSE YC7
1170 IF XS=XP THEN GOTO 1330
1180 REM POSSO CALCOLARE IL COEFFICIENTE ANGOLARE
1190 M=(YP-YS)/(XP-XS)
1200 IF ABS(M)>1 THEN M=1/M:GOTO 1380
1210 REM CALCOLA L'INTERCETTA ALL'ORIGINE PER UNA VARIAZIONE DELLE XC
1220 Q=YS-XS*M
1230 REM TRACCIO LA RETTA
1240 FOR XC=XP TO XS STEP SGN(XS-XP)
1250 YC=M*XC+Q:GOSUB1100:NEXT XC
1260 RETURN
1270 REM LA RETTA E' PARALLELA ALL'ASSE XC
1280 YC=YP
1290 FOR XC=XP TO XS STEP SGN(XS-XP)
1300 GOSUB1100:NEXT XC
1310 RETURN
1320 REM LA RETTA E' PARALLELA ALL'ASSE YC
1330 XC=XP
1340 FOR YC=YP TO YS STEP SGN(YS-YP)
1350 GOSUB1100:NEXT YC
1360 RETURN
1370 REM CALCOLO INTERCETTA ALL'ORIGINE PER UNA VARIAZIONE DELLE YC
1380 Q=XS-YS*M
1390 REM TRACCIO LA RETTA
1400 FOR YC=YP TO YS STEP SGN(YS-YP)
1410 XC=M*YC+Q:GOSUB1100:NEXT YC
1420 RETURN
1430 REM ROUTINE:TERMINA IL MODO VIDEO BIT-MAP
1440 REM SPENGO IL MODO BIT-MAP
1450 POKE VI+17,PEEK(VI+17) AND 223
1460 POKE VI+24,PEEK(VI+24) AND 247
```

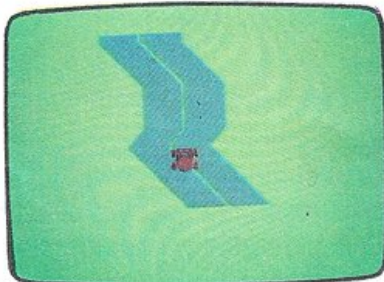
```
1470 RETURN
1490 REM ROUTINE:GESTISCE L'INPUT DI UNA STRINGA ALFANUMERICA
1500 REM LE VARIABILI UTILIZZATE SONO:Z6,Z7,Z8,Z9,Z8$,IN$
1510 REM IN INGRESSO VUOLE IL VALORE ZL CHE E' LA LUNGHEZZA MASSIMA
1520 REM DELLA STRINGA DA LEGGERE
1530 REM IN USCITA DA' LA STRINGA LETTA IN IN$ E LA SUA LUNGHEZZA IN Z9
1550 REM CANCELLO LA ZONA DI SCHERMO IN CUI VERRA' DIGITATA LA STRINGA
1570 FOR Z8=1 TO ZL: PRINT " ";:NEXT Z8
1580 FOR Z8=1 TO ZL: PRINT "█";:NEXT Z8
1590 IN$="":Z7=TI
1610 REM LEGGO UN CARATTERE
1630 GET Z8$:IF Z8$<>"" THEN 1730
1650 REM ACCENSIONE E SPEGNIMENTO DEL CURSORE
1670 IF Z7<TI AND NOT(Z6) THEN PRINT "██";:Z6=NOT(Z6):Z7=TI+15
1680 IF Z7<TI AND Z6 THEN PRINT " █";:Z6=NOT(Z6):Z7=TI+15
1690 GOTO 1630
1710 REM E' STATO DIGITATO UN CARATTERE
1730 Z8=ASC(Z8$):Z9=LEN(IN$)
1750 REM SE NON E' UN CARATTERE ALFANUMERICO, DEVE ESSERE UN RETURN O DELETE
1770 IF NOT((Z8>47 AND Z8<58) OR (Z8>64 AND Z8<91)) THEN GOTO 1890
1790 REM CONTROLLO CHE NON SIA STATA SUPERATA LA LUNGHEZZA MASSIMA
1810 IF Z9=ZL THEN GOTO 1630
1830 REM LO AGGIUNGO ALLA STRINGA IN$
1850 IN$=IN$+Z8$:PRINT Z8$;:GOTO 1630
1870 REM SE E' UN RETURN, HO TERMINATO LA LETTURA
1890 IF Z8=13 THEN PRINT " █";:RETURN
1910 REM SE E' DELETE, CANCELLO IN IN$ E SUL VIDEO L' ULTIMO CARATTERE DIGITATO
1930 IF Z8=20 AND Z9>0 THEN IN$=LEFT$(IN$,Z9-1):PRINT " ██";:GOTO 1630
1940 GOTO 1630
1960 REM ROUTINE:INIZIALIZZAZIONE COSTANTI
1980 REM CIRCUITO VIDEO
2000 VI=53248
2020 REM CIRCUITO SUONO
2040 SI=54272
2060 REM MEMORIA VIDEO
2080 MV=1024
2100 REM MEMORIA COLORE
2120 MC=55296
2140 REM COSTANTI DI USO COMUNE
2160 ZL=9
2180 REM INIZIALIZZAZIONE CHIP SUONO
2200 FOR I=0 TO 24
2210 POKE SI+I,0
2220 NEXT I
2230 RETURN
2250 REM ROUTINE:STAMPA IL TITOLO DEL PROGRAMMA MEMORIZZATO IN PG$
2270 GOSUB 2000
2280 POKE VI+32,15
2290 POKE VI+33,15
2300 PRINT "██████";
2310 PRINT TAB(6);"┌──────────────────────────┐"
2320 FOR I=1 TO 5
2330 PRINT TAB(6);"│                          │"
2340 NEXT I
2350 PRINT TAB(6);"└──────────────────────────┘"
2360 PRINT TAB(6);"████████DIGITA █RETURN█ PER PROSEGUIRE"
2370 PRINT "██████"
2380 FOR I=1 TO 5
2390 PRINT TAB(7);
2400 FOR J=1 TO 26
2410 PRINT "██ ";
2420 NEXT J
2430 PRINT
2440 NEXT I
2460 REM ORA SCRIVO IL TITOLO
2480 PRINT "█████████";TAB((40-LEN(PG$))/2);"██";PG$
2490 GET Z9$
2500 IF Z9$<>CHR$(13) THEN GOTO 2490
2510 PRINT "██";
2520 RETURN
```

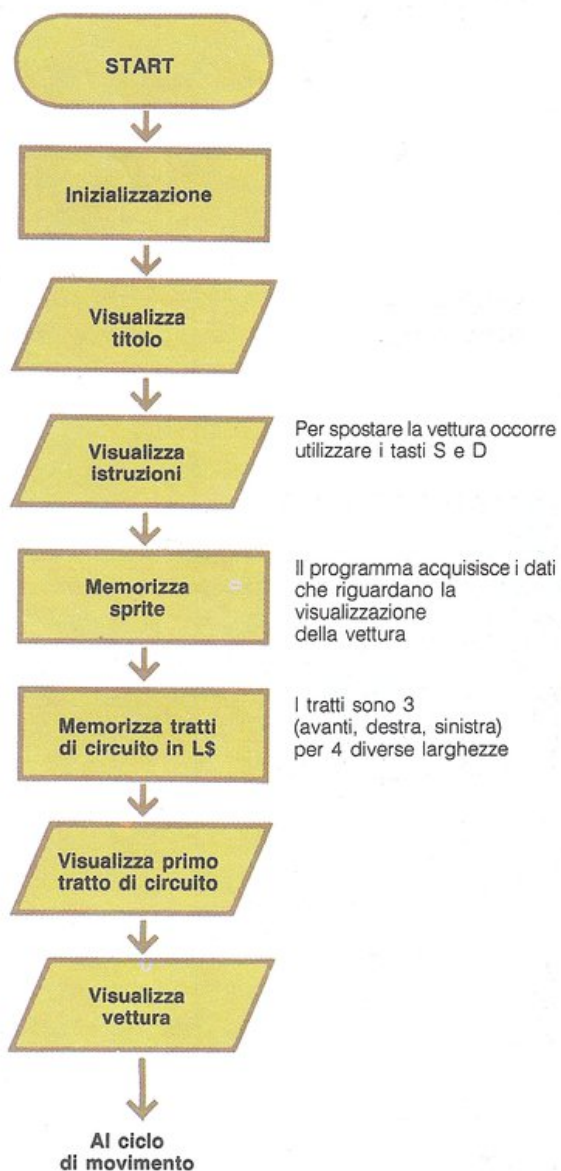
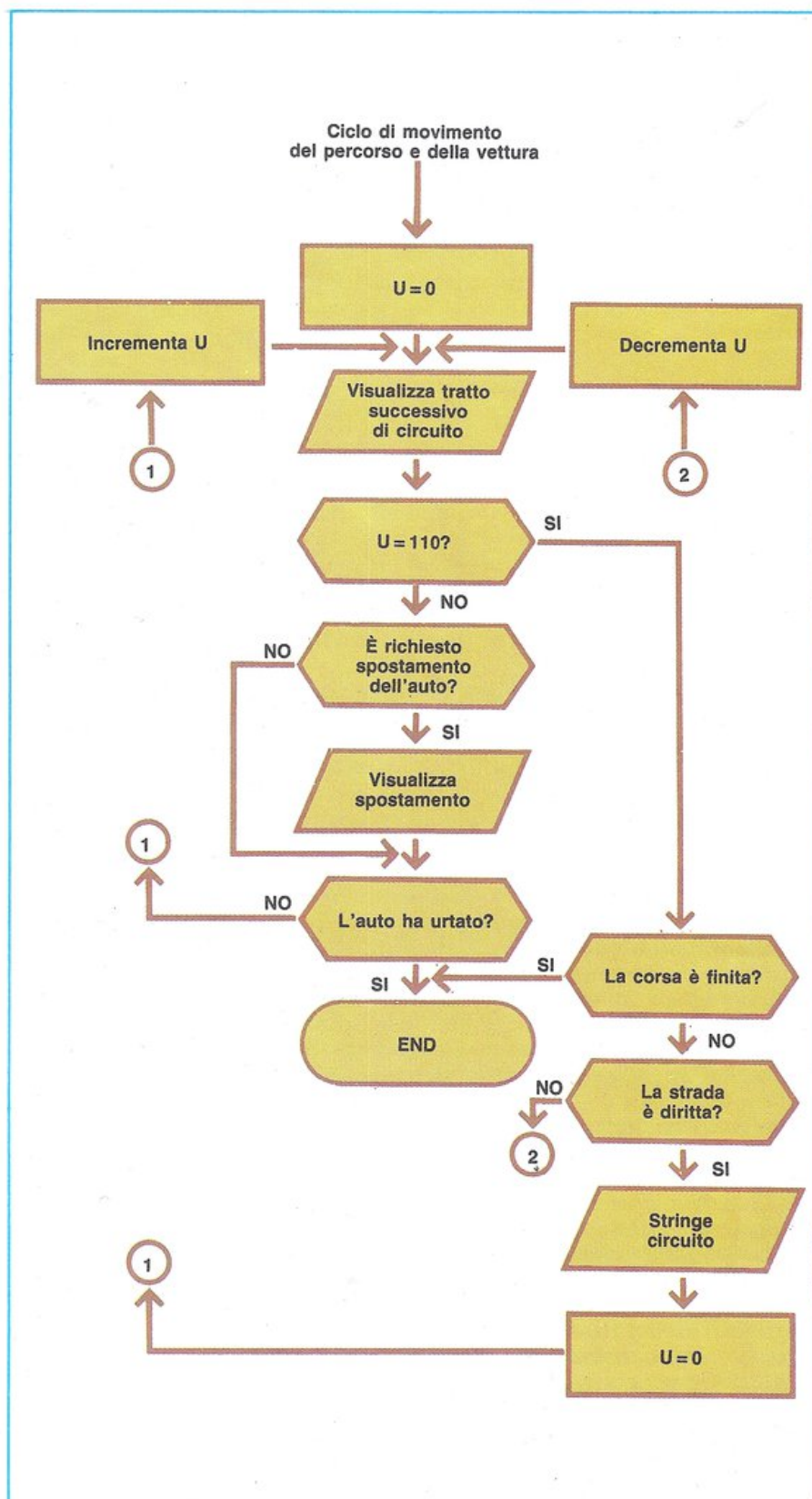
Corsa d'auto

Questo programma permetterà di simulare la corsa di un'automobile lungo un circuito. Avremo così la possibilità di esaminare un altro esempio di applicazione degli sprites, in cui dovremo inoltre gestire l'operazione di spostamento. Nel nostro caso lo spostamento avverrà solo orizzontalmente, in modo da permettere all'automobile di non uscire dalla pista che scorre sotto le sue ruote. Nel caso di uscita dal tracciato sarà simulato un urto utilizzando una tecnica tipica dei «cartoons». Oltre a cambiare direzione, la pista si restringerà progressivamente, fino a raggiungere una larghezza quasi pari a quella dell'auto. Dopo ogni urto il gioco sarà sospeso e il programma chiederà al giocatore se vuole effettuare un'altra partita.
Per rendere più interessante il gioco il programma costruirà per ogni partita un tracciato differente.

Analisi del problema

Ogni volta che vogliamo simulare il movimento di un oggetto sul video dobbiamo decidere, in base alle caratteristiche che desideriamo abbia il movimento e alle potenzialità del calcolatore, quali sono gli strumenti più adatti per ingannare l'occhio dell'osservatore. Nel nostro caso attueremo due strategie distinte: la marcia in avanti dell'auto sarà simulata tramite lo scorrimento verticale del circuito, mentre gli spostamenti orizzontali che permettono all'auto di seguire il tracciato saranno effettuati realmente.
Lo spostamento orizzontale sarà comandato dal giocatore con i tasti S (sinistra) e D (destra), mentre quello verticale sarà gestito, insieme alla forma del tracciato e al progressivo restringimento della pista col procedere della corsa, direttamente dal programma.
Per visualizzare e far scorrere il tracciato memorizzeremo i singoli tratti della pista in un array (vettore), il cui indice ci permetterà di leggere il tratto di circuito di larghezza e direzione desiderata. In questo caso l'array conterrà dodici elementi, in quanto abbiamo previsto per il circuito quattro larghezze e tre direzioni possibili (destra, sinistra, avanti); per disporre comunque di un numero maggiore o minore di possibilità di restringimento della sede stradale sarà sufficiente dimensionare un array più lungo o più corto.

Per dare l'impressione del movimento verticale della vettura sarà necessario visualizzare tratti susseguenti di circuito letti di volta in volta dall'array. Per ogni tratto disegnato sarà incrementato un contatore, il cui controllo permetterà di restringere la pista dopo averne visualizzato 110 tratti (righe dello schermo).
L'immagine della macchina sarà ottenuta tramite uno sprite, disegnato utilizzando i dati inseriti in apposite istruzioni DATA, che permetteranno di dargli forma e colore.
La gestione degli spostamenti orizzontali dell'auto sarà effettuata utilizzando i codici associati ai tasti S e D, che determineranno uno spostamento dello sprite di 8 bits (la larghezza di un carattere) a destra o a sinistra.
Il programma dovrà anche controllare se una ruota della vettura fuoriesce dal circuito, verificando se durante la corsa si è sovrapposta ad un punto dello sfondo: in tal caso, tramite un apposito ciclo di istruzioni POKE, farà lampeggiare il video per simulare l'urto e chiederà se si vuole giocare ancora. Il controllo sarà effettuato sulle ruote posteriori.
In ogni partita sarà presentato un tracciato differente; utilizzando opportunamente la funzione RND (che fornisce valori casuali compresi tra 0 e 1) il programma determinerà diversi valori dell'indice dell'array, e quindi sceglierà ogni volta un percorso di forma diversa.

Diagrammi di flusso

Il programma per prima cosa stampa il titolo, inizializza le costanti del C-64 e dimensiona le variabili. Visualizza poi le istruzioni per l'utente e inizializza un generatore di numeri casuali compresi tra 0 e 1. Segue la memorizzazione dello sprite che rappresenta la vettura e dei dati riguardanti la sua grandezza, il suo colore e la sua posizione iniziale. Nei due blocchi successivi sono memorizzati in L$ i quattro diversi «moduli» che compongono il circuito, ognuno nelle tre direzioni possibili (curva a destra, curva a sinistra, avanti dritto), è disegnato in grigio il primo tratto rettilineo, di lunghezza pari a 25 caratteri, ed è posizionata la vettura.

A questo punto ha inizio il gioco e il circuito comincia a scorrere cambiando direzione in modo casuale; coperti 110 tratti della pista (variabile U), il programma verifica se sono state utilizzate tutte e quattro le dimensioni previste: se sì, termina, altrimenti verifica se la direzione del circuito è diritta, nel qual caso può effettuare il restringimento della pista. Nel caso che ciò non fosse possibile, decrementa di una unità la variabile di controllo U e torna a disegnare un tratto di circuito con le dimensioni precedenti. Se invece sono stati visualizzati meno di 110 tratti di circuito, il programma verifica se è stato richiesto uno spostamento della vettura, e in caso affermativo lo esegue. Se si è avuto un urto, si fa lampeggiare il video e, dopo aver cancellato il disegno, si visualizza la richiesta di consenso a continuare. Per effettuare una nuova partita basterà digitare S ed immettere RETURN; altrimenti premendo N e digitando RETURN, il gioco terminerà.

Il programma

Le linee 290, 300 sono dedicate alla stampa del titolo e alla inizializzazione delle costanti del C-64 (routines 2710 e 2440). Visualizzati in blu lo sfondo dello schermo (linea 350) e in bianco le istruzioni per l'utente (linea 360÷390), il programma rimane in attesa che sia premuto il tasto che dà l'ordine di proseguire (linea 400). La linea 440 inizializza la variabile T, in cui è caricato il valore casuale ottenuto dalla funzione RND; la 480 stabilisce la dimensione dell'array L$ in cui saranno memorizzati i singoli tratti del circuito, e la 520 chiama la routine 2110, che visualizza e memorizza il disegno della vettura utilizzando i dati inseriti nelle istruzioni DATA delle linee 2350÷2380. Si colora di verde lo sfondo con le apposite istruzioni POKE (linea 560) e si stabiliscono le dimensioni dell'auto (linea 600); qualora si volesse raddoppiare la lunghezza dello sprite basterà cambiare l'istruzione POKE VI+23,0 con POKE VI+23,1. La linea 640 cancella l'ultima riga in basso dello schermo per eliminare l'antiestetica visualizzazione progressiva dei tratti del circuito, la 680 inizializza le variabili di gestione dello schermo, la 720 memorizza lo sprite nel tredicesimo blocco di memoria, la 760 lo colora in rosso e la 800 posiziona la ruota posteriore destra dell'auto

```
100 REM **************
110 REM *CORSA D'AUTO*
120 REM **************
140 REM VARIABILI UTILIZZATE
150 REM L$()          :VETTORE CONTENENTE I "MODULI" CHE COMPONGONO IL CIRCUITO
160 REM PG$           :MEMORIZZA IL TITOLO DEL PROGRAMMA
170 REM J,T           :CONTATORI DI CICLO
180 REM A$            :CARATTERE IN INGRESSO
190 REM I             :INDICE DEL VETTORE L$()
200 REM K             :POSIZIONE ORIZZONTALE DELLA VETTURA
210 REM B             :INDICA SE LA CORSA E' TERMINATA
220 REM A,C,N1,TM,SW:VARIABILI DI COMODO
230 REM U             :NUMERO DI SEGMENTI DI CIRCUITO VISUALIZZATI
240 REM N             :NUMERO DI SPAZI A DESTRA DEL CIRCUITO VISUALIZZATO
250 REM NO            :INDICA SE LA RISPOSTA E' STATA AFFERMATIVA O NEGATIVA
270 REM VISUALIZZO NOME PROGRAMMA ED INIZIALIZZO COSTANTI COMMODORE
290 PG$="CORSA D'AUTO"
300 GOSUB 2710
320 REM PRESENTAZIONE GIOCO
330 REM SFONDO BLU
350 POKE VI+32,6:POKE VI+33,6
360 PRINT "▒▒▒ATTENZIONE:"
370 PRINT "▒▒▒PER MUOVERE LA VETTURA "
380 PRINT "▒▒USARE I TASTI ▒S▒(SINISTRA) ▒D▒(DESTRA)"
390 PRINT "▒▒▒▒▒▒PER PROSEGUIRE PREMI ▒RETURN▒"
400 GET A$:IF A$="" THEN GOTO 400
420 REM INIZIALIZZO GENERATORE DI NUMERI CASUALI
440 T=RND(-TI)
460 REM DIMENSIONO ARRAY
480 DIM L$(12)
500 REM MEMORIZZO IL DISEGNO DELLA VETTURA
520 GOSUB 2110
540 REM SFONDO VERDE
560 POKE VI+32,5:POKE VI+33,5
580 REM LO SPRITE NON E' ESPANSO
600 POKE VI+23,0:POKE VI+29,0
620 REM RIDUCO IL VIDEO A 24 LINEE
640 POKE VI+17,PEEK(VI+17) AND 247 OR 7
660 REM INIZIALIZZO VARIABILI
680 M=0:K=163:U=0:I=0:B=0:C=1622
700 REM LO SPRITE E' MEMORIZZATO NEL 13-MO BLOCCO DA 64 BYTES
720 POKE 2040,13
740 REM IL COLORE DELLO SPRITE HA CODICE 2 (ROSSO)
760 POKE VI+39,2
780 REM POSIZIONO LO SPRITE CON LA RUOTA POSTERIORE DESTRA A CENTRO VIDEO
800 POKE VI,K:POKE VI+1,160
820 REM MEMORIZZO NEL VETTORE L$(), I TRATTI DI CIRCUITO DA VISUALIZZARE
840 L$(1)="◤     /     ◤"
850 L$(2)="▒    |    "
860 L$(3)="▌◥▒     \     ◥"
870 L$(4)="◤    /    ◤"
880 L$(5)="▒   |   "
890 L$(6)="▌◥▒    \    ◥"
900 L$(7)="◤   /   ◤"
910 L$(8)="▒  |  "
920 L$(9)="▌◥▒  \  ◥"
930 L$(10)="◤ / ◤"
940 L$(11)="▒ | "
950 L$(12)="▌◥▒ \ ◥"
970 REM DISEGNO IL PRIMO TRATTO (RETTILINEO) DEL CIRCUITO"
990 PRINT "▒"
1000 FOR N=1 TO 25
1010 PRINT TAB(11);"▒";L$(2)
1020 NEXT N
1040 REM IL FONDO STRADALE E' GRIGIO
1060 PRINT "▒";
```

al centro dello schermo. Alle linee 840÷950 sono memorizzati nel vettore L$ i tre moduli (curva a sinistra, avanti diritto, curva a destra) che compongono il circuito in ciascuna delle quattro larghezze previste (in tutto dodici elementi). Alle linee 1000÷1060 è disegnato il primo tratto del circuito, rettilineo e lungo 25 caratteri, è posizionato il ciglio sinistro del circuito ad una distanza pari ad 11 caratteri dal bordo sinistro del video ed è colorato in grigio il fondo stradale. Visualizzato lo sprite (linea 1110), ha inizio il ciclo principale che gestisce lo sviluppo (direzione e restringimento) del circuito e lo spostamento della vettura, e verifica se questa è uscita di strada (linee 1160÷1460). Visualizzato ciascun tratto del circuito, si incrementa il contatore U, che controlla il numero dei segmenti già tracciati. Qualora ne siano stati già tracciati 110 si passa alla linea 1550, che controlla la variabile booleana B (flag di fine corsa, che può assumere solo i valori 0 e 1). Se B=1 la corsa è terminata, altrimenti occorre controllare l'indice del vettore L$, per porre B uguale a 1 nel caso in cui tale indice abbia assunto il valore 9 (si è arrivati all'ultimo tratto del circuito). In questo caso si colorano gli ultimi 15 tratti della pista in bianco, altrimenti si prosegue, e se la direzione corrente del circuito è rettilinea (linea 1650), si restringe la strada (linee 1690÷1720); in caso contrario occorre aggiungere altri tratti al circuito fino a trovarne uno rettilineo, per poter

```
1070 N=11
1090 REM VISUALIZZO LA VETTURA
1110 POKE VI+21,1
1130 REM INIZIA IL CICLO PRINCIPALE DEL PROGRAMMA
1140 REM DISEGNO SULLA 25-MA RIGA UN TRATTO DI CIRCUITO ALTO UN CARATTERE
1160 PRINT SPC(N);L$(M+2+I)
1180 REM SE HO DISEGNATO 110 LINEE DEL CIRCUITO CON LA STESSA LARGHEZZA,
1190 REM LO RESTRINGO E CONTROLLO CHE LA CORSA NON SIA TERMINATA
1210 U=U+1:IF U=110 THEN GOTO 1550
1230 REM POSIZIONO LA VETTURA A SECONDA DELLO SPOSTAMENTO DESIDERATO
1250 POKE VI+16,SGN(K AND 256):POKE VI,K AND 255
1270 REM CONTROLLO SE E' RICHIESTO LO SPOSTAMENTO DELLA VETTURA
1290 A=PEEK(197)
1300 IF A=13 THEN K=K-8
1310 IF A=18 THEN K=K+8
1330 REM CONTROLLO SE LE RUOTE DELLA VETTURA SONO USCITE DAL
1340 REM CIGLIO STRADALE
1360 IF PEEK(C+K/8)=32 OR PEEK(C+1+K/8)=32 THEN GOTO 1770
1380 REM DECIDO SE E IN CHE VERSO IL CIRCUITO DEVE CAMBIARE DIREZIONE
1400 IF RND(1)>.7 THEN M=INT(RND(1)*3)-1
1420 REM CONTROLLO CHE IL CIRCUITO SIA TUTTO COMPRESO NELLA LINEA DI SCHERMO
1440 N1=N+M
1450 IF N1>28 OR N1<0 THEN N1=N:M=0
1460 N=N1
1480 REM RICOMINCIA IL CICLO PRINCIPALE
1500 GOTO 1160
1520 REM SE B VALE 1 ALLORA IL CIRCUITO E' TERMINATO (HO VISUALIZZATO
1530 REM 15 LINEE DI CIRCUITO IN BIANCO)
1550 IF B THEN GOTO 1960
1570 REM SE I VALE 9 SONO TERMINATI I TRATTI DI CIRCUITO DA VISUALIZZARE
1580 REM QUINDI PONGO B=1 E FACCIO DISEGNARE LE ULTIME 15 LINEE IN BIANCO
1600 IF I=9 THEN B=1:PRINT "[illegible]";:U=95:GOTO 1250
1620 REM NON POSSO STRINGERE LA PISTA SE LA SUA DIREZIONE NON E' RETTILINEA
1630 REM IN QUESTO MODO I TRATTI DI CIRCUITO NON SONO NECESSARIAMENTE LUNGHI 110
1650 IF M<>0 THEN U=109:GOTO 1250
1670 REM RESTRINGO LA PISTA
1690 N=N+1
1700 U=0
1710 M=0
1720 I=I+3
1730 GOTO 1250
1750 REM LA VETTURA E' USCITA FUORI PISTA. FACCIO LAMPEGGIARE IL VIDEO
1770 FOR N=1 TO 30
1780 POKE VI+32,1
1790 FOR T=1 TO 10
1800 NEXT T
1810 POKE VI+32,5
1820 NEXT N
1840 REM SPENGO LO SPRITE
1860 POKE VI+21,0
1880 REM AVVERTO L'UTENTE E GLI CHIEDO SE VUOLE GIOCARE ANCORA
1900 PRINT "[illegible]SEI ANDATO A SBATTERE!"
1910 GOTO 1990
1930 REM LA CORSA E' TERMINATA. L'UTENTE HA VINTO; LO AVVERTO E GLI CHIEDO
1940 REM SE VUOLE GIOCARE ANCORA
1960 FOR N=1 TO 1000:NEXT N
1970 POKE VI+21,0:REM CANCELLA LO SPRITE
1980 PRINT "[illegible]HAI FINITO LA CORSA!"
1990 PRINT
2010 REM RICHIESTA DI GIOCARE
2030 GOSUB 2190
2040 PRINT "[illegible]";
2050 IF SN THEN GOTO 680
2060 PRINT "[illegible]";:POKE VI+17,PEEK(VI+17) AND 251 OR 8
2070 POKE 53280,14:POKE 53281,6:END
```

finalmente eseguire il restringimento.
A questo punto il programma verifica se è stato richiesto uno spostamento orizzontale della vettura, nel qual caso lo effettua, a sinistra (linea 1300) o a destra (linea 1310); quindi controlla se una delle ruote posteriori della vettura si è venuta a trovare fuori del circuito (linea 1360): se sì, comanda il lampeggiamento dei contorni del video (linee 1770÷1820), la sparizione dell'auto (linea 1860) e, per ultimo, la visualizzazione del messaggio contenuto nella linea 1900.
La linea 2030 chiama la routine 2190, in cui si svolgono diverse operazioni: si chiede al giocatore se vuole effettuare un'altra partita, si gestisce la scelta della risposta, si memorizza nella variabile NO il tipo di risposta avuta (linee 2280, 2290), e quando è premuto il tasto RETURN (linea 2300) si passa il valore della variabile NO alla SN e lo si controlla, in modo da poter decidere se ricominciare il gioco o terminarlo (linea 2050). Qualora l'auto non sia uscita di strada, il programma decide, utilizzando la funzione RND (linea 1400), se e in che modo il circuito deve cambiare direzione. Nel caso in cui la nuova direzione scelta porti il circuito a sparire dalla visuale, il programma impone di proseguire dritto (linee 1440÷1460). Tramite l'istruzione della linea 1500 è nuovamente iniziato il ciclo principale (linee 1160÷1500).

```
2090 REM ROUTINE CHE MEMORIZZA IL DISEGNO DELLA VETTURA
2110 FOR N=64*13 TO 64*13+62
2120 READ A
2130 POKE N,A
2140 NEXT N
2150 RETURN
2170 REM ROUTINE CHE CHIEDE AL GIOCATORE DI RISPONDERE SI O NO
2190 NO=0
2200 TM=0
2210 SW=1
2220 PRINT "■GIOCHI ANCORA ? ■SI  NO■"
2230 IF TI<TM THEN GOTO 2270
2240 PRINT TAB(16+NO*4);MID$("■■",SW,1);MID$("SI NO ",3*NO+1,3);"■"
2250 SW=3-SW
2260 TM=TI+15
2270 GET D$
2280 IF D$="N" THEN NO=1:GOTO 2220
2290 IF D$="S" THEN NO=0:GOTO 2220
2300 IF D$=CHR$(13) THEN SN=(NO=0):RETURN
2310 GOTO 2230
2330 REM DATI RIGUARDANTI IL DISEGNO DELLA VETTURA
2350 DATA 0,0,0,71,255,226,239,60,247,254,24,127,253,255,191,237,129,183
2360 DATA 237,0,183,205,255,179,13,0,176,15,255,240,15,255,240,15,255,240
2370 DATA 15,255,240,239,255,247,255,255,255,255,255,255,255,255,255,246,255
2380 DATA 111,243,126,207,1,255,128,0,126,00
2400 REM ROUTINE DI INIZIALIZZAZIONE COSTANTI
2420 REM CIRCUITO VIDEO
2440 VI=53248
2460 REM CIRCUITO SUONO
2480 SI=54272
2500 REM MEMORIA VIDEO
2520 MV=1024
2540 REM MEMORIA COLORE
2560 MC=55296
2580 REM COSTANTI DI USO COMUNE
2600 ZL=9
2620 REM INIZIALIZZAZIONE CHIP SUONO
2640 FOR I=0 TO 24
2650 POKE SI+I,0
2660 NEXT I
2670 RETURN
2690 REM ROUTINE DI STAMPA DEL TITOLO DEL PROGRAMMA MEMORIZZATO IN PG$
2710 GOSUB 2440
2720 POKE VI+32,15
2730 POKE VI+33,15
2740 PRINT "■■■■";
2750 PRINT TAB(6);"┌──────────────────────────┐"
2760 FOR I=1 TO 5
2770 PRINT TAB(6);"│                          │"
2780 NEXT I
2790 PRINT TAB(6);"└──────────────────────────┘"
2800 PRINT TAB(6);"■■■■■■■DIGITA ■RETURN■ PER PROSEGUIRE"
2810 PRINT "■■■■■"
2820 FOR I=1 TO 5
2830 PRINT TAB(7);
2840 FOR J=1 TO 26
2850 PRINT "■■ ";
2860 NEXT J
2870 PRINT
2880 NEXT I
2900 REM ORA SCRIVO IL TITOLO
2920 PRINT "■■■■■■■■■";TAB((40-LEN(PG$))/2);"■■";PG$
2930 GET Z9$
2940 IF Z9$<>CHR$(13) THEN GOTO 2930
2950 PRINT "■■";
2960 RETURN
```

Alta risoluzione

Con questo programma si porterà il lettore a familiarizzare con alcuni degli strumenti grafici evoluti disponibili sul CBM-64. L'obiettivo è quello di affrontare i problemi che si presentano nel disegno di figure geometriche e nell'approssimazione di linee curve con segmenti di retta.
In particolare proveremo a disegnare delle spirali la cui apertura potrà variare, in funzione dei valori forniti in ingresso al programma, in modo da vedere gli effetti delle variazioni di questi valori sul disegno stesso.

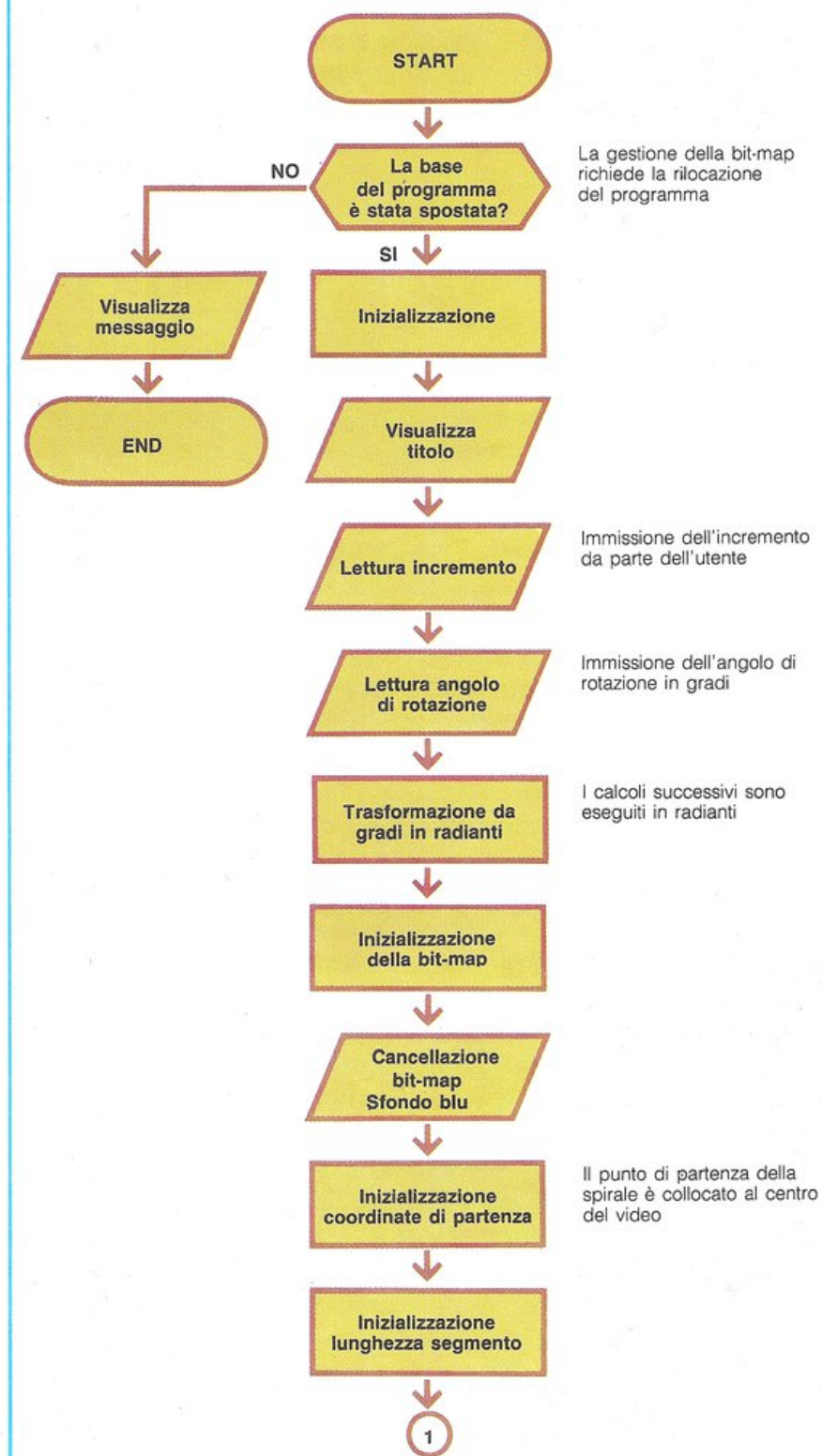
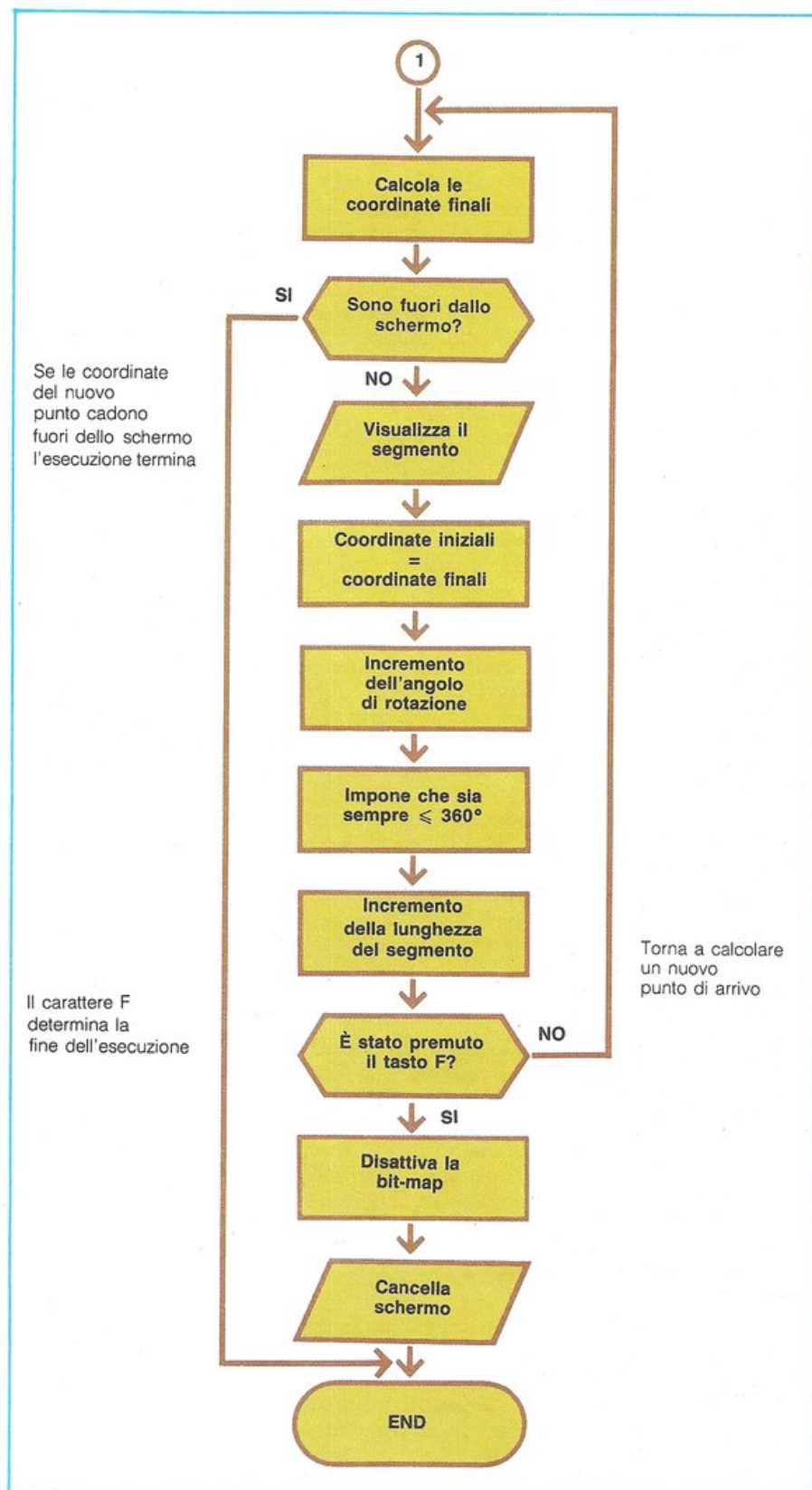
Analisi del problema

Usando gli strumenti del CBM-64 possiamo tracciare un disegno curvilineo in un solo modo: accendendo nella «scacchiera» rappresentata dal video i punti per i quali passa la curva. Poiché i punti sono in numero finito il risultato non sarà mai una curva perfetta, ma sarà sempre una approssimazione ottenuta con una successione di segmenti rettilinei che formano una spezzata. In questo programma studieremo le conseguenze di questa approssimazione attraverso il tracciamento di linee a spirale, costituite da tanti segmenti, ognuno ruotato di un certo angolo rispetto al precedente.
Per ovviare alla scarsa definizione del video in modalità testo abbiamo la possibilità di ricorrere alla bit-map, che ci permette una rappresentazione più dettagliata.
Passiamo quindi ad affrontare il problema di come si disegna la spezzata che approssima la spirale. Tracciando una successione di segmenti di lunghezza costante otterremmo il disegno di un poligono regolare; per evitare ciò dovremo quindi per prima cosa stabilire un incremento nella lunghezza dei successivi segmenti di retta che saranno visualizzati sullo schermo. Ovviamente, quanto maggiore sarà l'incremento tanto più la spirale risulterà «aperta». È necessario inoltre determinare l'angolo di rotazione di un segmento rispetto al precedente, che influenzerà l'apertura della figura.
Stabiliti gli strumenti necessari per la realizzazione del disegno, dobbiamo studiare come sia possibile realizzare ogni segmento che la compone. Immaginiamo lo schermo come un piano cartesiano, in modo da poter associare ad ogni singolo punto le rispettive coordinate. Possiamo calcolare le coordinate di ogni segmento, partendo da quelle dell'ultimo punto del segmento precedente, applicando le seguenti formule:

$$X_S = X_P + LU * COS\ (TG)$$
$$Y_S = Y_P + LU * SIN\ (TG)$$

dove X_S e Y_S sono le coordinate del punto di arrivo, X_P e Y_P le coordinate del punto di partenza, LU è la lunghezza del segmento, TG l'angolo di rotazione. Calcolato il punto di arrivo tracciamo il segmento che lo congiunge con quello di partenza: se il segmento risulterà parallelo ad uno dei due assi cartesiani, sarà possibile semplificare le operazioni di calcolo, e quindi determinare più velocemente tutti i punti del segmento da tracciare; se invece non risulterà parallelo a nessuno dei due assi cartesiani, si calcolerà di nuovo il coefficiente angolare (cioè l'inclinazione) del segmento che unisce il punto di partenza con quello di arrivo.
Individuati tutti i punti del segmento possiamo accenderli nella bit-map per ottenere la sua visualizzazione.
Porremo il punto di origine della figura al centro dello schermo, per avere la massima estensione del disegno.
Infine, per controllare che i valori forniti in ingresso per l'incremento e per l'angolo di rotazione siano numerici, li porremo in una stringa alfanumerica, che sarà trasformata in un valore numerico mediante la funzione VAL. Tale funzione restituirà il valore zero se la stringa letta contiene caratteri non numerici.

Diagrammi di flusso

Il primo blocco controlla se la base del programma è stata posta nella locazione di memoria 16384, al fine di permettere di collocare la bit-map dove normalmente risiede il programma.
Seguono l'inizializzazione delle costanti del C-64 e la visualizzazione del titolo. Sono poi richiesti l'incremento e l'angolo di rotazione in gradi sessagesimali, che l'utente deve impostare tramite la tastiera.
Subito dopo il programma trasforma il valore dei gradi sessagesimali in radianti, poiché le funzioni SIN e COS richiedono che il loro argomento sia espresso proprio in radianti. Si ha poi la fase di inizializzazione dello schermo in modo grafico, cancellando tutto ciò che è tracciato nella bit-map. Si devono quindi inizializzare le coordinate di partenza del segmento di retta, calcolando anche l'incremento.

Tale calcolo sarà ottenuto sommando alle coordinate del segmento di partenza la lunghezza del segmento incrementato moltiplicata per il coseno dell'angolo di rotazione per quanto riguarda l'asse delle X, e per il seno per l'asse delle Y. A questo punto è necessario verificare se il punto di arrivo calcolato cade all'esterno dello schermo. In caso negativo, il programma traccia un segmento di retta fra il punto di arrivo e quello di partenza e prosegue, altrimenti cessa l'esecuzione prima di tracciare il segmento.
Procedendo nell'elaborazione, si calcola l'incremento da attribuire all'angolo di rotazione: tale angolo deve essere sempre compreso fra 0 e 360°; se supera i 360° gli è sottratto il valore 2∗π (radianti), che corrisponde appunto a 360°. Successivamente si verifica se è stato premuto il tasto F.
L'immissione del carattere F determina la fine dell'esecuzione del programma, la disattivazione della bit-map e la cancellazione del video.
In caso contrario l'esecuzione continua a partire dal calcolo delle coordinate di arrivo, per determinare l'indirizzo di un nuovo punto.

Il programma

All'inizio dell'esecuzione la linea 300 controlla se la base del programma è stata rilocata all'indirizzo di memoria 16384. Se non è così si visualizzano le istruzioni necessarie (linee 320÷400). Dopo l'inizializzazione delle costanti e la visualizzazione del titolo (linee 430, 440), le linee 460, 470 chiedono il valore con cui incrementare il segmento di retta e chiamano la routine 1570, che cancella quella parte dello schermo su cui saranno visualizzati i valori dell'incremento del segmento e dell'angolo inseriti dall'utente. Segue la lettura dei caratteri immessi controllando che non sia superata la lunghezza massima di 5 caratteri (sia per l'incremento sia per l'angolo sessagesimale). La linea 480 trasforma il valore della stringa in valore numerico e lo attribuisce alla variabile IC (incremento del segmento). Se il valore dell'incremento del segmento è minore o uguale a 0 (linea 490) allora si torna alla linea 460. Alle linee 510, 520 è visualizzata la richiesta del valore dell'angolo sessagesimale. La linea 530 chiama nuovamente la routine di gestione della stringa alfanumerica, per memorizzare i valori desiderati per l'angolo di rotazione (GS), ripetendo le operazioni già effettuate per l'incremento del segmento, con la differenza che il valore dell'angolo di

```
100 REM ****************
110 REM ALTA RISOLUZIONE
120 REM ****************
140 REM VARIABILI UTILIZZATE
160 REM I,J   :CONTATORI DI CICLO
170 REM IC    :INCREMENTO DELLA LUNGHEZZA DEL SEGMENTO
180 REM GS    :GRADI SESSAGESIMALI
190 REM GR    :GRADI RADIANTI
200 REM TG    :ANGOLO DI ROTAZIONE DEL SEGMENTO
210 REM CO    :COLORE DELLO SFONDO DELLA BIT-MAP
220 REM XP,YP:COORDINATE DEL PUNTO DI PARTENZA DEL SEGMENTO
230 REM XS,YS:COORDINATE DEL PUNTO DI ARRIVO DEL SEGMENTO
240 REM LU    :LUNGHEZZA CORRENTE DEL SEGMENTO
250 REM XC,YC:COORDINATE DEL PUNTO DELLA BIT-MAP DA ACCENDERE
260 REM A$    :VARIABILE DI INGRESSO
280 REM CONTROLLO CHE LA BASE DEL PROGRAMMA SIA STATA MESSA ALLA LOCAZIONE 16384
300 IF PEEK(44)=64 THEN GOTO 430
310 REM AVVERTO CHE IL PROGRAMMA NON PUO' ANDARE IN ESECUZIONE
320 PRINT "▒▒▒▒▒"
330 PRINT "ATTENZIONE, HAI DIMENTICATO"
340 PRINT "▒▒DI IMMETTERE I SEGUENTI COMANDI:"
350 PRINT "▒▒▒1) POKE 44,64"
360 PRINT "▒▒2) POKE 16384,0"
370 PRINT "▒▒PRIMA DI IMMETTERLI RICORDA"
380 PRINT "▒▒DI SALVARE IL PROGRAMMA,SE QUESTO"
390 PRINT "▒NON E' GIA' PRESENTE SU NASTRO,"
400 PRINT "▒E POI DI RICARICARLO"
410 END
420 REM STAMPA TITOLO ED INIZIALIZZAZIONE COSTANTI C64
430 PG$="ALTA RISOLUZIONE"
440 GOSUB 2270
450 REM VALORE DI CUI INCREMENTARE LA LUNGHEZZA DEL SEGMENTO
460 PRINT "▒▒▒▒▒▒PER GIOCARE ANCORA DIGITA 'G'▒"
462 PRINT "▒▒▒PER TERMINARE DIGITA 'F'▒"
465 PRINT "▒▒▒▒▒INCREMENTO PER IL SEGMENTO ? ";
470 ZL=5:GOSUB 1570:PRINT
480 IC=VAL(IN$)
490 IF IC<=0 THEN GOTO 460
500 REM ANGOLO DI ROTAZIONE
510 PRINT "▒▒ANGOLO DI ROTAZIONE, IN GRADI "
520 PRINT "▒SESSAGESIMALI ? ";
530 ZL=5:GOSUB 1570:PRINT
540 GS=VAL(IN$)
550 IF GS=0 THEN PRINT "▒                    ▒":GOTO 520
560 REM TRASFORMO I GRADI SESSAGESIMALI IN GRADI RADIANTI
570 GR=GS*π/180:TG=GR
580 REM INIZIO IL DISEGNO
590 REM INIZIALIZZO GRAFICA
600 GOSUB 950
610 REM CANCELLO IL VIDEO BIT-MAP
620 GOSUB 1040
630 REM COLORO IN BLU
640 CO=6:GOSUB 1070
650 REM POSIZIONI DI INIZIO
660 XP=160
670 YP=100
680 LU=0
690 REM CALCOLO COORDINATE DEL PUNTO DI ARRIVO
700 XS=XP+LU*COS(TG)
710 YS=YP+LU*SIN(TG)
720 REM SE UNO DEI PUNTI DEL SEGMENTO E' FUORI QUADRO,SMETTO DI DISEGNARE
730 IF XS>319 OR YS>199 THEN GOTO 870
740 REM DISEGNO IL SEGMENTO
750 GOSUB 1150
760 XP=XS
770 YP=YS
780 TG=TG+GR
790 TG=TG-INT(TG/2/π)*2*π
800 LU=LU+IC
```

rotazione può anche essere minore di 0.
*Alla linea 570 si trasformano i gradi sessagesimali in radianti (gradi radianti = gradi sessagesimali * π/180), quindi la linea 600 chiama la routine 950, cui è demandato il compito di inizializzare la bit-map dello schermo. La bit-map (BM) è caricata in memoria a partire dall'indirizzo 8192, riservandole uno spazio di 8 kbytes. La routine gestisce ogni singolo punto della bit-map partendo dal primo punto in alto a sinistra.*
La linea 620 chiama la routine 1040 che cancella l'eventuale immagine presente e la linea 640 chiama la routine 1070, che colora il fondo della bit-map in blu.
Alle linee 660÷680 il punto di partenza del segmento (XP, YP) è posizionato al centro dello schermo ed è inizializzata la variabile LU (lunghezza del segmento).
Le linee 700, 710 calcolano le coordinate del punto di arrivo del segmento (XS, YS) sommando alle coordinate del punto di partenza (XP, YP) la lunghezza del segmento data dall'utente moltiplicata per il coseno dell'angolo di rotazione (per quanto concerne l'asse X) o per il seno dello stesso angolo (per quanto concerne l'asse delle Y).
La linea 730 verifica se le coordinate dei punti di arrivo cadono fuori schermo: in questo caso si ferma l'esecuzione e il disegno termina nel punto di arrivo dell'ultimo segmento tracciato.
Di seguito, la linea 750 chiama la routine 1150 che traccia il segmento dal punto di partenza al punto di arrivo. Se la coordinata

```
810 REM SE E' PREMUTO IL TASTO 'E', TERMINO IL DISEGNO
820 GET A$ IF A$="F" THEN GOTO 840
825 IF A$="G" THEN GOSUB 1450 : GOTO 460
829 GOTO 700
830 REM TERMINA IL DISEGNO
840 GOSUB 1450
850 PRINT "[CLR]"
860 POKE VI+32,14:POKE VI+33,6:END
870 GET A$ IF A$="" THEN GOTO 870
880 IF A$="F" THEN GOTO 840
885 IF A$="G" THEN GOSUB 1450 : GOTO 460
887 GOTO 840
890 REM **********************************************************************
900 REM I SOTTOPROGRAMMI PER LA GESTIONE DELLA GRAFICA, SONO SCRITTI
910 REM IN MODO DA RENDERE IL PIU' VELOCE POSSIBILE LA LORO AZIONE
920 REM **********************************************************************
930 REM ROUTINE INIZIALIZZA IL MODO VIDEO BIT-MAP E LE VARIABILI INTERESSATE
940 REM ATTIVO IL MODO BIT-MAP
950 POKE VI+17,PEEK(VI+17) OR 32
960 REM LA MEMORIA BIT-MAP E' ALLA LOCAZIONE 8192
970 POKE VI+24,PEEK(VI+24) OR 8
980 BM=8192:Z0=8:Z1=320:Z2=7:Z3=1/Z0
990 FOR I=0 TO 7
1000 Z3(I)=2↑I
1010 NEXT I
1020 RETURN
1030 REM ROUTINE:CANCELLO LA MEMORIA BIT-MAP
1040 Z9=0:FORI=BMTOBM+7999:POKEI,Z9:NEXT
1050 RETURN
1060 REM ROUTINE:IMPOSTO LO SFONDO NEL COLORE CO
1070 FORI=1024TO2023:POKEI,CO:NEXT
1080 RETURN
1090 REM ROUTINE:ACCENDE IL PUNTO DI COORDINATE XC,YC
1100 BY=BM+INT(YC*Z3)*Z1+INT(XC*Z3)*Z0+(YCANDZ2)
1110 POKEBY,PEEK(BY)ORZ3(Z2-(XCANDZ2))
1120 RETURN
1130 REM ROUTINE:TRACCIA UNA RETTA DA XP,YP A XS,YS
1140 REM LA RETTA E' PARALLELA ALL'ASSE XC?
1150 IF YP=YS THEN GOTO 1280
1160 REM LA RETTA E' PARALLELA ALL'ASSE YC?
1170 IF XS=XP THEN GOTO 1330
1180 REM POSSO CALCOLARE IL COEFFICIENTE ANGOLARE
1190 M=(YP-YS)/(XP-XS)
1200 IF ABS(M)>1 THEN M=1/M:GOTO 1380
1210 REM CALCOLO INTERCETTA ALL'ORIGINE PER UNA VARIAZIONE DELLE XC
1220 Q=YS-XS*M
1230 REM TRACCIO LA RETTA
1240 FOR XC=XP TO XS STEP SGN(XS-XP)
1250 YC=M*XC+Q:GOSUB1100:NEXT
1260 RETURN
1270 REM LA RETTA E' PARALLELA ALL'ASSE XC
1280 YC=YP
1290 FOR XC=XP TO XS STEP SGN(XS-XP)
1300 GOSUB1100:NEXT
1310 RETURN
1320 REM LA RETTA E' PARALLELA ALL'ASSE YC
1330 XC=XP
1340 FOR YC=YP TO YS STEP SGN(YS-YP)
1350 GOSUB1100:NEXT
1360 RETURN
1370 REM CALCOLO INTERCETTA ALL'ORIGINE PER UNA VARIAZIONE DELLE YC
1380 Q=XS-YS*M
1390 REM TRACCIO LA RETTA
1400 FOR YC=YP TO YS STEP SGN(YS-YP)
1410 XC=M*YC+Q:GOSUB1100:NEXT
1420 RETURN
1430 REM ROUTINE:TERMINA IL MODO VIDEO BIT-MAP
1440 REM SPENGO IL MODO BIT-MAP
1450 POKE VI+17,PEEK(VI+17) AND 223
```

Y del punto di partenza è uguale a quella del punto d'arrivo (YS=YP, linee 1280, 1290) il segmento risulta parallelo all'asse X; di conseguenza si calcolano le sole coordinate X di tutti i punti del segmento che unisce il punto di partenza a quello di arrivo, e si chiama la routine 1100, che visualizza il segmento. Se la coordinata X del punto di partenza è uguale a quella del punto d'arrivo, e quindi se il segmento è parallelo all'asse Y (XS=XP, linea 1170), si eseguono le medesime operazioni (linea 1330) del caso precedente, invertendo la X con la Y. Se infine il segmento non è parallelo né all'asse X né all'asse Y allora si calcola il coefficiente angolare (cioè l'inclinazione) del segmento di retta che unisce il punto di partenza al punto di arrivo (linea 1190), e si traccia il segmento così determinato, per tornare poi alla linea 750. Alle linee 760, 770 il programma assegna le coordinate del punto di arrivo al nuovo punto di partenza, poi calcola il nuovo angolo di rotazione del segmento (linee 780, 790), sommando al precedente (TG) i gradi radianti prestabiliti per il successivo (calcolati alla linea 570). Qualora l'angolo ottenuto superi l'angolo giro (360 gradi = 2∗π radianti) verrà utilizzato l'angolo ottenuto sottraendo all'angolo di rotazione l'angolo giro. La linea 800 calcola la lunghezza del segmento successivo sommando l'incremento al precedente valore della lunghezza.

```
1460 POKE VI+24,PEEK(VI+24) AND 247
1470 RETURN
1490 REM ROUTINE:GESTISCE L'INPUT DI UNA STRINGA ALFANUMERICA
1500 REM LE VARIABILI UTILIZZATE SONO:Z6,Z7,Z8,Z9,Z8$,IN$
1510 REM IN INGRESSO VUOLE IL VALORE ZL CHE E' LA LUNGHEZZA MASSIMA
1520 REM DELLA STRINGA DA LEGGERE
1530 REM IN USCITA DA' LA STRINGA LETTA IN IN$ E LA SUA LUNGHEZZA IN Z9
1550 REM CANCELLO LA ZONA DI SCHERMO IN CUI VERRA' DIGITATA LA STRINGA
1570 FOR Z8=1 TO ZL: PRINT " ";:NEXT Z8
1580 FOR Z8=1 TO ZL: PRINT "▌";:NEXT Z8
1590 IN$="":Z7=TI
1610 REM LEGGO UN CARATTERE
1630 GET Z8$:IF Z8$<>"" THEN 1730
1650 REM ACCENSIONE E SPEGNIMENTO DEL CURSORE
1670 IF Z7<TI AND NOT(Z6) THEN PRINT "▒▌";:Z6=NOT(Z6):Z7=TI+15
1680 IF Z7<TI AND Z6 THEN PRINT " ▌";:Z6=NOT(Z6):Z7=TI+15
1690 GOTO 1630
1710 REM E' STATO DIGITATO UN CARATTERE
1730 Z8=ASC(Z8$):Z9=LEN(IN$)
1750 REM SE NON E' UN CARATTERE ALFANUMERICO, DEVE ESSERE UN RETURN O DELETE
1770 IF NOT(Z8>31 AND Z8<96) THEN GOTO 1890
1790 REM CONTROLLO CHE NON SIA STATA SUPERATA LA LUNGHEZZA MASSIMA
1810 IF Z9=ZL THEN GOTO 1630
1830 REM LO AGGIUNGO ALLA STRINGA IN$
1850 IN$=IN$+Z8$:PRINT Z8$;:GOTO 1630
1870 REM SE E' UN RETURN, HO TERMINATO LA LETTURA
1890 IF Z8=13 THEN PRINT " ▌";:RETURN
1910 REM SE E' DELETE, CANCELLO IN IN$ E SUL VIDEO L' ULTIMO CARATTERE DIGITATO
1930 IF Z8=20 AND Z9>0 THEN IN$=LEFT$(IN$,Z9-1):PRINT " ▌▌";:GOTO 1630
1940 GOTO 1630
1960 REM ROUTINE:INIZIALIZZAZIONE COSTANTI
1980 REM CIRCUITO VIDEO
2000 VI=53248
2020 REM CIRCUITO SUONO
2040 SI=54272
2060 REM MEMORIA VIDEO
2080 MV=1024
2100 REM MEMORIA COLORE
2120 MC=55296
2140 REM COSTANTI DI USO COMUNE
2160 ZL=9
2180 REM INIZIALIZZAZIONE CHIP SUONO
2200 FOR I=0 TO 24
2210 POKE SI+I,0
2220 NEXT I
2230 RETURN
2250 REM ROUTINE:STAMPA IL TITOLO DEL PROGRAMMA MEMORIZZATO IN PG$
2270 GOSUB 2000
2280 POKE VI+32,15
2290 POKE VI+33,15
2300 PRINT "▒▒▒▒▒▒";
2310 PRINT TAB(6);"┌──────────────────────────┐"
2320 FOR I=1 TO 5
2330 PRINT TAB(6);"│                          │"
2340 NEXT I
2350 PRINT TAB(6);"└──────────────────────────┘"
2360 PRINT TAB(6);"▒▒▒▒▒▒▒▒DIGITA ▒RETURN▒ PER PROSEGUIRE"
2370 PRINT "▒▒▒▒▒▒"
2380 FOR I=1 TO 5
2390 PRINT TAB(7);
2400 FOR J=1 TO 26
2410 PRINT "▒▒ ";
2420 NEXT J
2430 PRINT
2440 NEXT I
2460 REM ORA SCRIVO IL TITOLO
2480 PRINT "▒▒▒▒▒▒▒▒▒▒";TAB((40-LEN(PG$))/2);"▒▒";PG$
2490 GET Z9$
2500 IF Z9$<>CHR$(13) THEN GOTO 2490
2510 PRINT "▒▒";
2520 RETURN
```

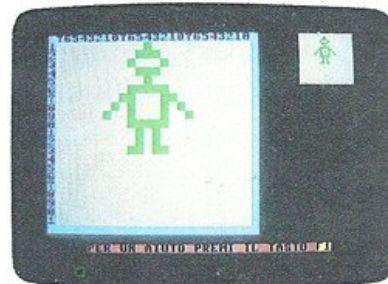
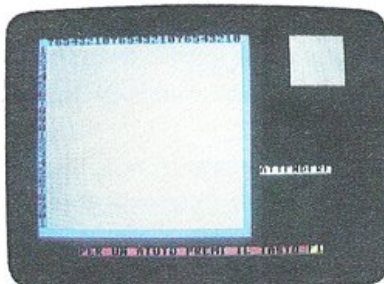
Disegna il tuo sprite

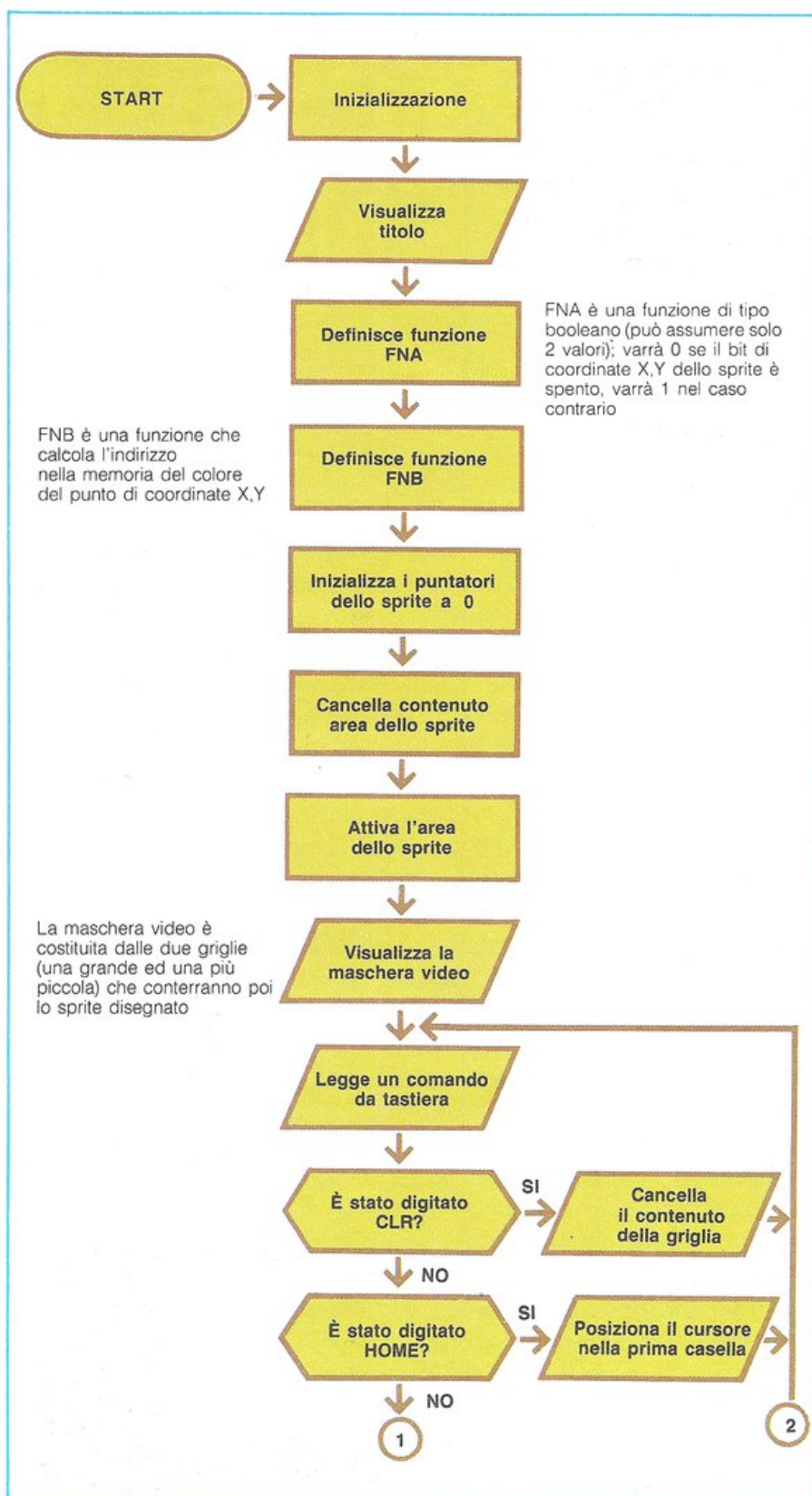
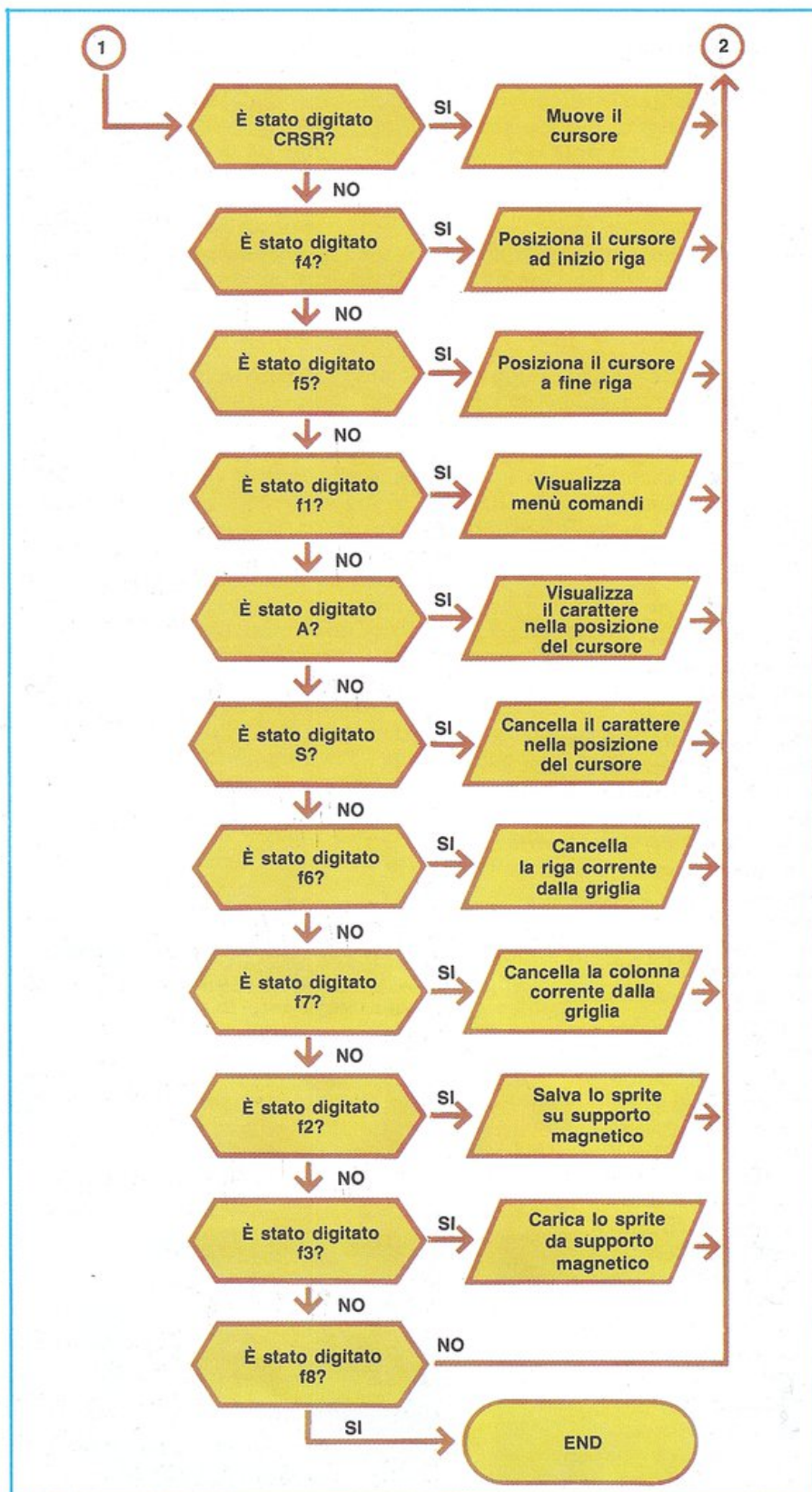
Il CBM-64 consente l'uso di elementi grafici definiti dall'utente (sprites). Il disegno degli sprites è una delle operazioni più lunghe e noiose: è infatti necessario determinare manualmente i valori dei 64 bytes utilizzati per ciascuno di essi, trasformare i 64 gruppi di otto bits in 64 caratteri da introdurre nel programma mediante le istruzioni DATA e, infine, farli leggere al programma.
Risulterà interessante a questo scopo realizzare un programma di utilità che consenta di definire facilmente gli sprites, disegnandoli direttamente sul video per poi memorizzarli eventualmente in un file.

Analisi del problema

Per facilitare il lavoro di tracciamento dello sprite, sarà consentito all'utente di disegnarlo in una porzione del video ingrandita (griglia); contemporaneamente sarà utilizzata un'altra parte del video per visualizzare lo sprite in dimensioni naturali.
Per disegnare lo sprite nella griglia più grande, l'utente potrà spostare il cursore e accendere o spegnere la casella in cui si trova il cursore stesso. Gli spostamenti del cursore saranno possibili attivando i tasti CRSR, mentre l'accensione e lo spegnimento della casella saranno realizzati attraverso altri due tasti, ad esempio A ed S.
Sarà inoltre necessario prevedere i comandi di gestione, come ad esempio la memorizzazione dello sprite, il suo richiamo in memoria per modificarlo ed infine l'uscita dal programma. Poiché questi comandi sono numerosi, sarà opportuno raggrupparli in un menù, che potrà essere richiamato in un momento qualsiasi dell'esecuzione.
Il programma avrà la struttura tipica degli editor (programmi per la gestione di testi, immagini, ecc.) per cui, dopo aver visualizzato le due griglie, entrerà in un ciclo costituente il corpo del programma stesso, in cui leggerà un comando qualunque, verificherà se il comando è legale, lo eseguirà e tornerà a leggere un nuovo comando da tastiera.
I comandi potranno essere di diversi tipi:

- spostamento del cursore
- accensione o spegnimento di una casella
- cancellazione di una riga o di una colonna della griglia
- memorizzazione dello sprite su un file esterno
- caricamento di uno sprite da un file esterno
- visualizzazione del menù comandi.

La realizzazione del programma dovrà inoltre prevedere alcuni controlli, quale ad esempio quello che serve per evitare lo spostamento del cursore fuori della griglia. Sarà infine necessario prevedere, durante il passaggio dal menù comandi alle due griglie, un «rinfresco» del video, per ridisegnare lo sprite su entrambe le griglie.

Diagrammi di flusso

Il programma comincia con l'inizializzazione delle costanti e con la visualizzazione del titolo. Successivamente definisce la funzione FNA, che ha il compito di verificare se un determinato bit dello sprite è acceso o spento, e la funzione FNB, utilizzata per calcolare l'indirizzo di memoria dei punti (coordinate X e Y) che serviranno per visualizzare lo sprite. Il blocco successivo cancella la griglia più piccola, riservata alla definizione dello sprite, e la riaccende dopo aver messo a 0 tutti i bits. Il programma deve poi acquisire, tramite l'istruzione GET, un comando fornitogli da tastiera e confrontarlo con un certo numero di istruzioni, per stabilire ciò che deve essere eseguito:
— CLR: cancellare lo schermo;
— HOME: posizionare il cursore nella prima casella in alto a sinistra;

— CRSR: eseguire gli spostamenti del cursore;
— f4: posizionare il cursore all'inizio della riga corrente;
— f5: posizionare il cursore alla fine della riga corrente;
— f1: visualizzare il menù comandi;
— A: visualizzare un rettangolo verde nella griglia grande e accendere il punto corrispondente nella griglia piccola;
— S: cancellare un rettangolo già tracciato;
— f6: cancellare la riga su cui è posizionato il cursore;
— f7: cancellare la colonna su cui è posizionato il cursore;
— f2: memorizzare lo sprite su supporto magnetico;
— f3: caricare lo sprite dal supporto magnetico;
— f8: terminare l'esecuzione del programma.
Se infine il codice immesso non rientra fra quelli previsti, il programma deve restare in attesa di un comando legale.

Il programma

Dopo la visualizzazione del titolo e l'inizializzazione delle costanti (linee 280, 290), le linee 310, 360 definiscono due funzioni: la FNA, di tipo booleano, e la FNB, che calcola l'indirizzo nella mappa dei colori (MC) corrispondente alla posizione del cursore sullo schermo. La linea 380 chiama la routine 1600, che svolge le seguenti operazioni:
— inizializza il circuito video, i puntatori allo sprite e i registri di memoria di posizione (linee 1600÷1620);
— cancella eventuali sprites rimasti in memoria da un'esecuzione precedente (linea 1640);
— stabilisce la lunghezza (linea 1660) e la larghezza (linea 1700) dello sprite;
— lo rappresenta in un solo colore (linea 1680 e linea 1740);
— lo carica nel tredicesimo blocco della memoria (linea 1760).

Tutte queste assegnazioni sono effettuate con istruzioni POKE che utilizzano la costante VI, contenente l'indirizzo del registro di gestione degli sprites: sommando ogni volta a VI un determinato valore si accede all'indirizzo del registro di gestione desiderato. Ad esempio, POKE VI+39,5 permette di accedere al registro di memoria cui è demandata l'assegnazione del colore dello sprite (VI+39), imponendo il colore 5 (verde).

La linea 400 chiama la routine 1090, che cancella il contenuto della zona riservata allo sprite, mentre

```
100 REM **********************
110 REM DISEGNA IL TUO SPRITE
120 REM **********************
140 REM VARIABILI UTILIZZATE
160 REM I,J       :CONTATORI DI CICLO
170 REM BY        :POSIZIONE DEL BYTE CONTENENTE IL BIT DA ACCENDERE O SPEGNERE
180 REM BI        :POSIZIONE NEL BYTE DEL BIT DA ACCENDERE O SPEGNERE
190 REM ST        :STATO DEL C64 DURANTE LA LETTURA DEI FILES
200 REM X,Y       :COORDINATE DEL PUNTO DA ACCENDERE SU VIDEO
210 REM X1,Y1     :VARIABILI DI COMODO
220 REM C$        :COMANDO LETTO DA TASTIERA
230 REM CO        :COLORE DEL PUNTO DA VISUALIZZARE; VERDE O GRIGIO O GIALLO
240 REM A         :VARIABILE PER L'INGRESSO DATI DA DISCO
250 REM FG        :=1 SE IL CURSORE E' IN ULTIMA COLONNA  (LA PIU' A DESTRA)
270 REM PRESENTAZIONE PROGRAMMA E INIZIALIZZAZIONE COSTANTI
280 PG$="DISEGNA IL TUO SPRITE"
290 GOSUB 3760
300 REM DEFINISCO LA FUNZIONE FNA DI TIPO BOOLEANO
310 REM QUESTA FUNZIONE VALE 0 SE IL BIT DELLO SPRITE CON COORDINATE X,Y E'
315 REM SPENTO, VALE 1 IN CASO CONTRARIO
320 DEF FNA(Z)=PEEK(832+(X-1)*3+INT((Y-1)/8)) AND 2↑(8-Y+INT((Y-1)/8)*8)
340 REM DEFINISCO UNA FUNZIONE FNB CHE CALCOLA L'INDIRIZZO IN MEMORIA COLORE
350 REM DEL PUNTO DI COORDINATE X,Y
360 DEF FNB(Z)=MC+40*X+Y
370 REM INIZIALIZZO CIRCUITO VIDEO E PUNTATORI ALLO SPRITE
380 GOSUB 1600
390 REM CANCELLO L'AREA RISERVATA ALLO SPRITE
400 GOSUB 1090
410 REM VISUALIZZO LO SPRITE NEI DUE FORMATI PREVISTI
420 GOSUB 1790
430 REM ACCENDO LO SPRITE
440 POKE VI+21,1
450 REM INIZIA IL PROGRAMMA
460 REM INIZIALIZZO COORDINATE CURSORE
470 X=1:Y=1
480 REM DISEGNO IL CURSORE NEL PUNTO DI COORDINATE X,Y
490 POKE FNB(Z),7
500 REM LEGGO COMANDI
510 GET C$:IF C$="" THEN 510
520 REM ESEGUO I COMANDI
530 REM CLR SPRITE
540 IF C$="■" THEN GOSUB 1090:GOSUB 2240
550 REM HOME
560 IF C$="■" THEN GOSUB 2240
570 REM DOWN
580 IF C$="■" THEN GOSUB 2280
590 REM UP
600 IF C$="■" THEN GOSUB 2330
610 REM RIGHT
620 IF C$="■" THEN GOSUB 2380
630 REM LEFT
640 IF C$="■" THEN GOSUB 2430
650 REM F4:CURSORE A INIZIO LINEA
660 IF C$="■" THEN GOSUB 2480
670 REM F5:CURSORE A FINE LINEA
680 IF C$="■" THEN GOSUB 2520
690 REM F1:HELP
700 IF C$="■" THEN GOSUB 890:GOSUB 1790:GOSUB 2020:GOTO 470
710 REM ACCENDE IL PUNTO
720 IF C$="A" THEN GOSUB 2580
730 REM SPEGNE IL PUNTO
740 IF C$="S" THEN GOSUB 2700
750 REM F6:CANCELLA LA LINEA CORRENTE
760 IF C$="■" THEN GOSUB 2800
770 REM F7:CANCELLA LA COLONNA CORRENTE
780 IF C$="■" THEN GOSUB 2880
790 REM F2:SALVA LO SPRITE SU DISCO
800 IF C$="■" THEN GOSUB 1160
810 REM F3:CARICA LO SPRITE DA DISCO
820 IF C$="■" THEN GOSUB 1300:GOSUB 2020:GOTO 470
830 REM F8:TERMINA LA SESSIONE
840 IF C$="■" THEN GOTO 1540
850 GOTO 490
860 REM INIZIANO LE ROUTINES
870 REM ROUTINE:MENU DI HELP
880 REM CAMBIO COLORE SFONDO
890 POKE VI+32,5:POKE VI+33,5
900 PRINT "■■";
910 PRINT TAB(14);"MENU COMANDI"
920 PRINT "■■I COMANDI: CLR, HOME E I TASTI"
930 PRINT "DI CONTROLLO CURSORE, HANNO "
940 PRINT "SIGNIFICATO INVARIATO■"
950 PRINT "F1";TAB(4);"VISUALIZZA QUESTA MASCHERA"
960 PRINT "F2";TAB(4);"SALVA LO SPRITE SU DISCO"
970 PRINT "F3";TAB(4);"LOAD LO SPRITE DA DISCO"
980 PRINT "F4";TAB(4);"CURSORE A INIZIO LINEA"
990 PRINT "F5";TAB(4);"CURSORE A FINE LINEA"
1000 PRINT "F6";TAB(4);"CANCELLA LA LINEA CORRENTE"
1010 PRINT "F7";TAB(4);"CANCELLA LA COLONNA CORRENTE"
1020 PRINT "F8";TAB(4);"TERMINA LA SESSIONE"
```

la 420 chiama un'altra routine, la 1790, che cancella tutto lo schermo, assegna allo sfondo il colore nero (linea 1810), disegna la griglia più piccola in grigio (linee 1830÷1850), disegna la griglia più grande con il suo contorno, ne numera due lati (linee 1870÷1920), e infine visualizza un'offerta di aiuto scritta in nero su sfondo rosso (linee 1940÷1990).
Le linee 470÷490 posizionano il cursore nella prima casella in alto a sinistra dello sprite. Alla linea 510 il C-64 si pone in attesa di un comando da tastiera. Le linee 540÷850 interpretano i comandi immessi da tastiera, chiamando, per ogni comando ricevuto, la routine che gestisce l'esecuzione di tale comando. In particolare il tasto CLR (linea 540) cancella il video (routine 1090) e posiziona il cursore sul primo carattere in alto a sinistra (routine 2240); il tasto HOME (linea 560) fa tornare il cursore sul primo carattere in alto a sinistra (routine 2240), il tasto CRSR↑ (linea 600) lo sposta di una riga in alto (routine 2330), il tasto CRSR↓ di una riga in basso (routine 2280), il tasto CRSR→ (linea 620) di una posizione verso destra (routine 2380), il tasto CRSR← (linea 640) di una posizione verso sinistra (routine 2430), il tasto f4 (linea 660) lo posiziona all'inizio della riga corrente (routine 2480) e il tasto f5 (linea 680) alla fine della riga corrente (routine 2520).
Il tasto f1 (linea 700) che visualizza il menù comandi, merita una trattazione più dettagliata,

```
1030 PRINT "A";TAB(4);"ACCENDE IL PUNTO SOTTO IL CURSORE"
1040 PRINT "S";TAB(4);"SPEGNE IL PUNTO SOTTO IL CURSORE"
1050 PRINT "██      PREMI UN TASTO"
1060 GET C$:IF C$="" THEN 1060
1070 RETURN
1080 REM ROUTINE:CANCELLA SPRITE
1090 FOR I=0 TO 62
1100 POKE 832+I,0
1110 NEXT I
1120 REM CANCELLO LA GRIGLIA
1130 GOSUB 1790
1140 RETURN
1150 REM ROUTINE:SALVA SPRITE SU DISCO
1160 PRINT "▓▓▓▓▓▓▓▓▓▓▓▓▓▓";SPC(27);"▓▓      SAVE     ■"
1170 PRINT SPC(27);"▓▓ NOME SPRITE?■▓▓■■■■■■■■■■■■■";
1180 ZL=10:GOSUB 3060
1190 IF IN$="" THEN GOTO 1180
1200 OPEN 5,8,5,"@0:"+IN$+",S,W"
1210 FOR I=0 TO 62
1220 PRINT#5,PEEK(832+I)
1230 NEXT I
1240 CLOSE 5
1250 REM CANCELLO LA SCRITTA PRECEDENTE
1260 PRINT "▓▓▓▓▓▓▓▓▓▓▓▓▓▓";SPC(27);"■              "
1270 PRINT SPC(27);"                    ▓▓■■■■■■■■■■■■■           "
1280 RETURN
1290 REM ROUTINE:CARICA LO SPRITE DA DISCO
1300 PRINT "▓▓▓▓▓▓▓▓▓▓▓▓▓▓";SPC(27);"▓▓      LOAD     ■"
1310 PRINT SPC(27);"▓▓ NOME SPRITE?■▓▓■■■■■■■■■■■■■";
1320 ZL=10:GOSUB 3060
1330 IF IN$="" THEN GOTO 1320
1340 OPEN 5,8,5,IN$+",S,R"
1350 I=0
1360 INPUT#5,A
1370 IF (ST=0) OR (ST=64) THEN GOTO 1450
1380 PRINT "◘▓ATTENZIONE:ERRORE LETTURA FILE ";IN$;"■"
1390 OPEN 15,8,15,"I"
1400 CLOSE 15
1410 CLOSE 5
1420 FOR I=1 TO 1000:NEXT I
1430 GOSUB 1790
1440 RETURN
1450 POKE 832+I,A
1460 I=I+1
1470 IF I<=62 THEN GOTO 1360
1480 CLOSE 5
1490 REM CANCELLO LA SCRITTA PRECEDENTE
1500 PRINT "▓▓▓▓▓▓▓▓▓▓▓▓▓▓";SPC(27);"■              "
1510 PRINT SPC(27);"                    ▓▓■■■■■■■■■■■■■           "
1520 RETURN
1530 REM FINE SESSIONE DI LAVORO
1540 POKE VI+21,0
1545 POKE VI+32,14
1550 POKE VI+33,6
1560 PRINT "◘▓";
1570 END
1580 REM INIZIALIZZO VIC-II E PUNTATORI ALLO SPRITE
1590 REM REGISTRI POSIZIONE SPRITE #1
1600 POKE VI,20
1610 POKE VI+1,53
1620 POKE VI+16,1
1630 REM SPENGO LO SPRITE
1640 POKE VI+21,0
1650 REM ESPANSIONE VERTICALE
1660 POKE VI+23,1
1670 REM NON E' MULTICOLORE
1680 POKE VI+28,0
1690 REM ESPANSIONE ORIZZONTALE
1700 POKE VI+29,1
1710 REM LO SPRITE HA PRIORITA' PIU' ELEVATA DELLO SFONDO
1720 POKE VI+27,0
1730 REM LO SPRITE E' VERDE
1740 POKE VI+39,5
1750 REM MEMORIZZO LO SPRITE NEL 13-MO BLOCCO DA 64 BYTES
1760 POKE 2040,13
1770 RETURN
1780 REM VISUALIZZO LO SPRITE NEI DUE FORMATI PREVISTI
1790 PRINT "◘";
1800 REM COLORE SFONDO IN NERO
1810 POKE VI+32,0:POKE VI+33,0
1820 REM DISEGNO IL CONTORNO DELLO SPRITE PICCOLO
1830 FOR I=1 TO 6
1840 PRINT SPC(31);"▓▓▓        ■"
1850 NEXT I
1860 REM DISEGNO LA GRIGLIA GRANDE
1870 PRINT "▓";
1880 PRINT "▓◘ 765432107654321076543210 "
1890 FOR I=1 TO 21
1900 PRINT "▓◘";RIGHT$(STR$(I-INT(I/10)*10),1);"▓                        ◘ "
1910 NEXT I
```

in quanto è gestito da una serie di routines: la 890 cambia il colore dello sfondo in verde, visualizza il menù e rimane in attesa di un comando inviatogli da un qualsiasi tasto per tornare alle griglie (linea 1060). La 1790 gestisce la visualizzazione delle maschere, come già descritto alla linea 400. La 2020, visualizzata la scritta ATTENDERE in nero su fondo bianco, riporta lo sprite eventualmente presente sulla griglia piccola in quella grande. Ciò si esegue tramite due cicli FOR che scandiscono ogni punto della griglia piccola e verificano, tramite la funzione FNA, se tale bit è acceso o spento; qualora si trovi il bit acceso lo si riporta in verde sulla griglia grande, mentre se lo si trova spento lo si riporta in grigio (linee 2050÷2130). Dopo il doppio ciclo si cancella il messaggio ATTENDERE (linea 2150) e, infine, si posiziona il cursore in alto a sinistra. Il tasto A (linea 720) sposta il cursore verso destra accendendo in colore verde il carattere precedentemente occupato dal cursore (routine 2580, che individua il byte e il bit corrispondenti al punto che bisogna accendere nella griglia piccola, linee 2580÷2630). Il tasto S (linea 740) cancella il carattere su cui è posizionato il cursore (routine 2700, che opera similmente alla 2580, con la differenza che spegne il bit interessato anziché accenderlo). Il tasto f6 (linea 760) cancella l'intera riga su cui è posizionato il cursore (routine 2800, che dopo aver cancellato tutti i caratteri della riga chiama la 2700, che spegne i punti

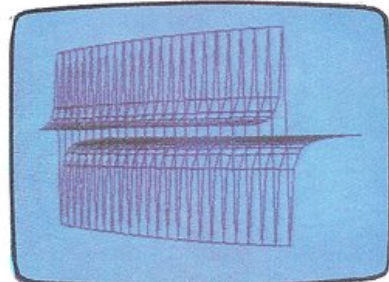
```
1920 PRINT "▒▒                                  "
1930 REM VISUALIZZO UNA SCRITTA DI AIUTO
1940 PRINT "▒";
1950 FOR I=1 TO 24
1960 PRINT
1970 NEXT I
1980 REM SONO POSIZIONATO SUL'ULTIMA RIGA
1990 PRINT "       ▒▒PER UN AIUTO PREMI IL TASTO ▒F1";
2000 RETURN
2010 REM ROUTINE CHE COPIA LO SPRITE SULLA GRIGLIA
2020 PRINT "▒":PRINT "▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒";SPC(27);"▒▒ATTENDERE"
2030 REM VISUALIZZO UN MESSAGGIO DI ATTESA
2040 REM SE IL BIT DI COORDINATE X,Y E' ACCESO,DISEGNO IN VERDE IL CARATTERE
2050 CO=15
2060 FOR X=1 TO 21
2070 FOR Y=1 TO 24
2080 REM SE IL BIT DI COORDINATE X,Y E' ACCESO,DISEGNO IN VERDE IL CARATTERE
2090 REM RELATIVO SULLA GRIGLIA; IN CASO CONTRARIO LO DISEGNO IN GRIGIO
2100 CO=15
2110 IF FNA(Z)>0 THEN CO=5
2120 POKE FNB(Z),CO
2130 NEXT Y,X
2140 REM CANCELLO IL MESSAGGIO PRECEDENTE
2150 PRINT "▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒";SPC(27);"▒           "
2160 RETURN
2170 REM ROUTINE CHE RIPRISTINA IL CARATTERE SOTTO IL CURSORE
2180 CO=15
2190 IF FNA(Z)>0 THEN CO=5
2200 POKE FNB(Z),CO
2210 RETURN
2220 REM ROUTINE CHE ESEGUE L'OPERAZIONE HOME
2230 REM RIPRISTINO IL CARATTERE SOTTO IL CURSORE
2240 GOSUB 2180
2250 X=1:Y=1
2260 RETURN
2270 REM ROUTINE CHE SPOSTA IL CURSORE IN BASSO DI UNA LINEA
2280 GOSUB 2180
2290 X=X+1
2300 IF X>21 THEN X=21
2310 RETURN
2320 REM ROUTINE CHE SPOSTA IL CURSORE IN ALTO DI UNA LINEA
2330 GOSUB 2180
2340 X=X-1
2350 IF X<1 THEN X=1
2360 RETURN
2370 REM ROUTINE CHE SPOSTA IL CURSORE A DESTRA
2380 GOSUB 2180
2390 Y=Y+1
2400 IF Y>24 THEN Y=24
2410 RETURN
2420 REM ROUTINE CHE SPOSTA IL CURSORE A SINISTRA
2430 GOSUB 2180
2440 Y=Y-1
2450 IF Y<1 THEN Y=1
2460 RETURN
2470 REM ROUTINE CHE SPOSTA IL CURSORE A INIZIO LINEA
2480 GOSUB 2180
2490 Y=1
2500 RETURN
2510 REM ROUTINE CHE SPOSTA IL CURSORE A FINE LINEA
2520 GOSUB 2180
2530 Y=24
2540 RETURN
2550 REM ROUTINE CHE ACCENDE IL PUNTO SOTTO IL CURSORE E SPOSTA IL CURSORE A
2560 REM DESTRA (SE E' POSSIBILE).
2570 REM ACCENDO IL BIT NELLA ZONA DI MEMORIA CONTENENTE LO SPRITE
2580 BY=(X-1)*3+INT((Y-1)/8)
2590 REM HO CALCOLATO IL BYTE INTERESSATO
2600 REM CALCOLO IL BIT INTERESSATO
2610 BI=8-Y+INT((Y-1)/8)*8
2620 REM ACCENDO IL PUNTO
2630 POKE 832+BY,PEEK(832+BY) OR 2↑BI
2640 REM CHIAMO LA ROUTINE CHE SPOSTA IL CURSORE A DESTRA
2650 GOSUB 2380
2660 RETURN
2670 REM ROUTINE CHE SPEGNE IL PUNTO SOTTO IL CURSORE E SPOSTA IL CURSORE A
2680 REM DESTRA (SE E' POSSIBILE).
2690 REM SPENGO IL BIT NELLA ZONA DI MEMORIA CONTENENTE LO SPRITE
2700 BY=(X-1)*3+INT((Y-1)/8)
2710 REM HO CALCOLATO IL BYTE INTERESSATO
2720 REM CALCOLO IL BIT INTERESSATO
2730 BI=8-Y+INT((Y-1)/8)*8
2740 REM SPENGO IL PUNTO
2750 POKE 832+BY,PEEK(832+BY) AND (255-2↑BI)
2760 REM CHIAMO LA ROUTINE CHE SPOSTA IL CURSORE A DESTRA
2770 GOSUB 2380
2780 RETURN
2790 REM ROUTINE CHE CANCELLA LA LINEA CORRENTE
2800 FOR Y1=1 TO 24
2810 Y=Y1
```

corrispondenti nella griglia piccola). Il tasto f7 (linea 780) cancella la colonna dove è posizionato il cursore (routine 2280, che cancella tutti i caratteri avvalendosi delle coordinate X anziché delle coordinate Y utilizzate nella routine 2800). Il tasto f2 (linea 800) memorizza lo sprite disegnato su supporto magnetico utilizzando la routine 1160, che svolge le seguenti operazioni: visualizza i messaggi SAVE e NOME SPRITE?, stabilisce tramite le costanti ZL la lunghezza massima di 10 caratteri alfanumerici per il nome che l'utente può attribuire allo sprite, chiama la routine 3060, e verifica che sia stato inserito almeno un carattere. La routine 3060, chiamata dalla 1160 (linea 1180), cancella la zona dello schermo dove sarà inserito il nome dello sprite, legge i caratteri inseriti (linea 3120) facendo lampeggiare il cursore (linee 3160, 3170) e memorizza i caratteri in una stringa alfanumerica, controllando la lunghezza di quest'ultima. I caratteri non alfanumerici non sono accettati (linea 3260), ad eccezione del DELETE e del RETURN. Il tasto f3 (linea 820) richiama da supporto magnetico l'eventuale sprite precedentemente memorizzato, utilizzando due routines: la 1300, che visualizza i messaggi LOAD e NOME SPRITE?, e la 2020, che visualizza il messaggio ATTENDERE, esegue la scansione dei punti come precedentemente illustrato e riposiziona il cursore in alto a sinistra. Il tasto f8 (linea 830) fa terminare l'esecuzione del programma chiamando la linea 1540.

```
2820 REM CANCELLO IL CARATTERE
2830 GOSUB 2700
2840 NEXT Y1
2850 Y=1
2860 RETURN
2870 REM CHE CANCELLA LA COLONNA CORRENTE
2880 FG=0:IF Y=24 THEN FG=1
2890 FOR X1=1 TO 21
2900 X=X1
2910 REM CANCELLO IL CARATTERE
2920 GOSUB 2700
2930 REM PORTO IL CURSORE A SINISTRA DI UN POSTO
2940 IF FG=0 THEN Y=Y-1
2950 NEXT X1
2960 RETURN
2980 REM ROUTINE CHE GESTSCE L'INPUT DI UNA STRINGA ALFANUMERICA.
2990 REM LE VARIABILI UTILIZZATE SONO:Z6,Z7,Z8,Z9,Z8$,IN$
3000 REM IN INGRESSO VUOLE LA LUNGHEZZA MASSIMA ZL
3010 REM DELLA STRINGA DA LEGGERE
3020 REM IN USCITA DA' IN IN$ LA STRINGA LETTA E IN Z9 LA SUA LUNGHEZZA
3040 REM CANCELLO LA ZONA DI SCHERMO IN CUI VERRA' DIGITATA LA STRINGA
3060 FOR Z8=1 TO ZL: PRINT " ";:NEXT Z8
3070 FOR Z8=1 TO ZL: PRINT "▌";:NEXT Z8
3080 IN$="":Z7=TI
3100 REM LEGGO UN CARATTERE
3120 GET Z8$:IF Z8$<>"" THEN 3220
3140 REM ACCENSIONE E SPEGNIMENTO DEL CURSORE
3160 IF Z7<TI AND NOT(Z6) THEN PRINT "▒▌";:Z6=NOT(Z6):Z7=TI+15
3170 IF Z7<TI AND Z6 THEN PRINT " ▌";:Z6=NOT(Z6):Z7=TI+15
3180 GOTO 3120
3200 REM E' STATO DIGITATO UN CARATTERE
3220 Z8=ASC(Z8$):Z9=LEN(IN$)
3240 REM SE NON E' UN CARATTERE ALFANUMERICO, DEVE ESSERE UN RETURN O DELETE
3260 IF NOT((Z8>47 AND Z8<58) OR (Z8>64 AND Z8<91)) THEN GOTO 3380
3280 REM CONTROLLO CHE NON SIA STATA SUPERATA LA LUNGHEZZA MASSIMA
3300 IF Z9=ZL THEN GOTO 3120
3320 REM LO AGGIUNGO ALLA STRINGA IN$
3340 IN$=IN$+Z8$:PRINT Z8$;:GOTO 3120
3360 REM SE E' UN RETURN, HO TERMINATO LA LETTURA
3380 IF Z8=13 THEN PRINT " ▌";:RETURN
3400 REM SE E' DELETE, CANCELLO IN IN$ E SUL VIDEO L' ULTIMO CARATTERE DIGITATO
3420 IF Z8=20 AND Z9>0 THEN IN$=LEFT$(IN$,Z9-1):PRINT " ▌▌";:GOTO 3120
3430 GOTO 3120
3450 REM ROUTINE DI INIZIALIZZAZIONE COSTANTI
3470 REM CIRCUITO VIDEO
3490 VI=53248
3510 REM CIRCUITO SUONO
3530 SI=54272
3550 REM MEMORIA VIDEO
3570 MV=1024
3590 REM MEMORIA COLORE
3610 MC=55296
3630 REM COSTANTI DI USO COMUNE
3650 ZL=9
3670 REM INIZIALIZZAZIONE CHIP SUONO
3690 FOR I=0 TO 24
3700 POKE SI+I,0
3710 NEXT I
3720 RETURN
3740 REM ROUTINE CHE STAMPA IL TITOLO DEL PROGRAMMA MEMORIZZATO IN PG$
3760 GOSUB 3490
3770 POKE VI+32,15
3780 POKE VI+33,15
3790 PRINT "▒▒▒▒▒";
3800 PRINT TAB(6);"┌──────────────────────────┐"
3810 FOR I=1 TO 5
3820 PRINT TAB(6);"|                          |"
3830 NEXT I
3840 PRINT TAB(6);"└──────────────────────────┘"
3850 PRINT TAB(6);"▒▒▒▒▒▒▒DIGITA ▒RETURN▒ PER PROSEGUIRE"
3860 PRINT "▒▒▒▒▒"
3870 FOR I=1 TO 5
3880 PRINT TAB(7);
3890 FOR J=1 TO 26
3900 PRINT "▒▒ ";
3910 NEXT J
3920 PRINT
3930 NEXT I
3950 REM ORA SCRIVO IL TITOLO
3970 PRINT "▒▒▒▒▒▒▒▒▒";TAB((40-LEN(PG$))/2);"▒▒";PG$
3980 GET Z9$
3990 IF Z9$<>CHR$(13) THEN GOTO 3980
4000 PRINT "▒▒";
4010 RETURN
```

Grafica tridimensionale

Questo programma consentirà di disegnare il grafico di una funzione di due variabili. Il disegno sembrerà una figura tridimensionale, sebbene sia visualizzato su di un piano. Per rappresentare il grafico di una funzione è necessario disporre di un numero finito di valori che la funzione assume in un dato intervallo. A questo scopo l'utente dovrà inserire i valori che determinano il campo di definizione della funzione, cioè il massimo e il minimo valore che si desidera assegnare alle due variabili indipendenti. Sarà necessario fornire anche il numero dei punti che si vogliono prendere in esame, il che permetterà di ottenere una maggiore o minore definizione del tracciato.
Molto interessante è la possibilità di disegnare più grafici di una stessa funzione considerata in diversi intervalli, che permetterà di confrontare i differenti andamenti che la funzione assume a seconda dell'intervallo dei valori assegnati alle variabili indipendenti.

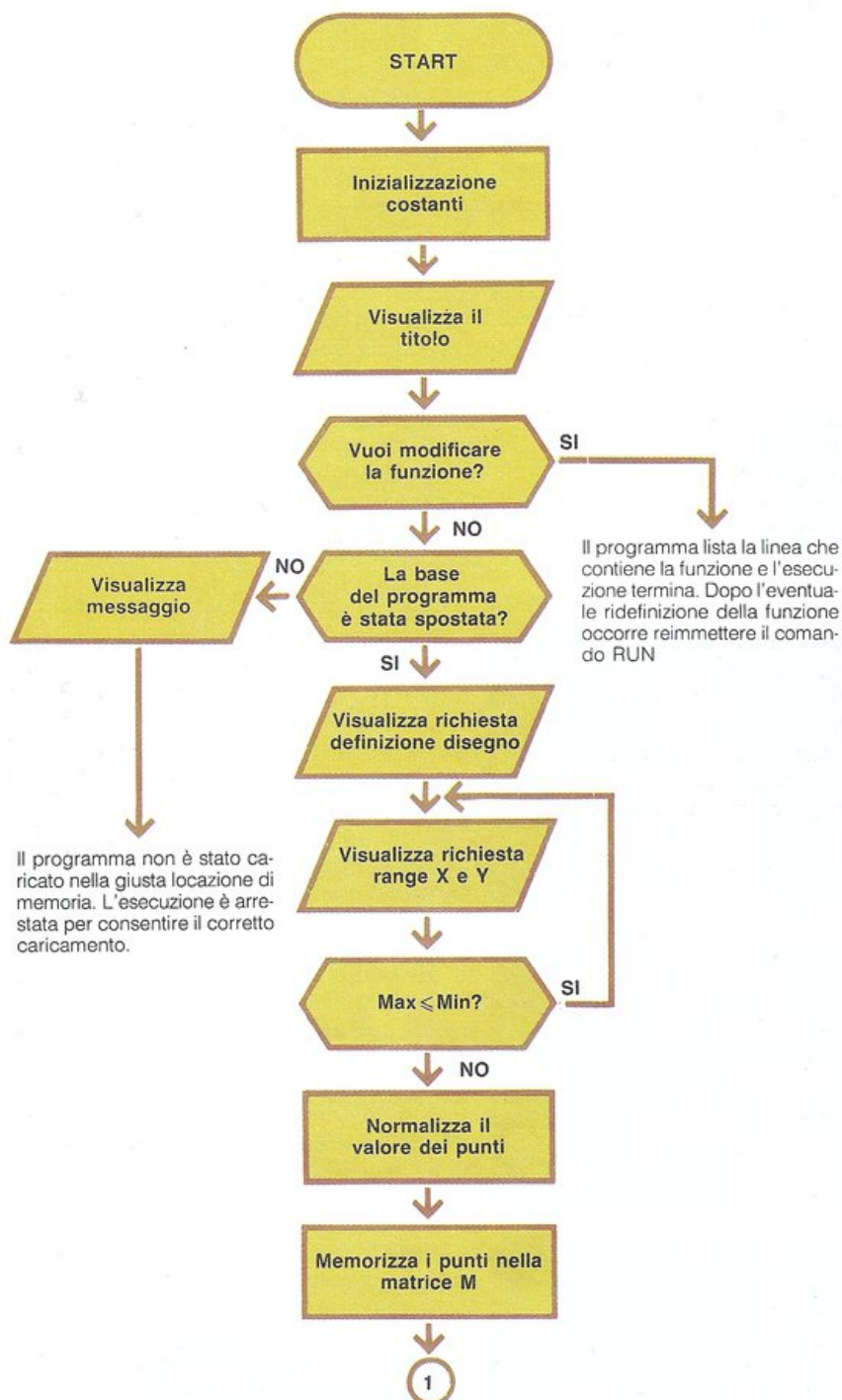
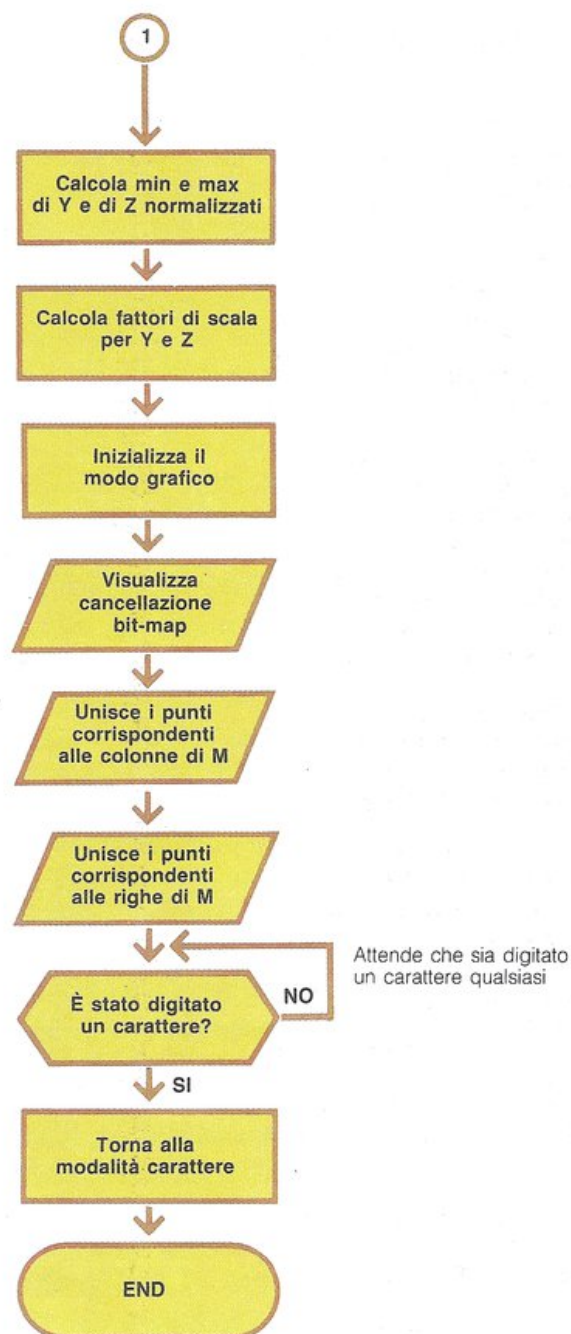
Analisi del problema

Ogni volta che ci troviamo a dover disegnare una figura tridimensionale su un piano ricorriamo alle regole della prospettiva, un artificio che permette di ingannare l'occhio di un osservatore dandogli l'impressione di vedere una figura tridimensionale dove realmente ne esiste solo una bidimensionale. Per tracciare grafici che abbiano una buona prospettiva anche il nostro programma ricorrerà ad un artificio simile.
Per poter disegnare il grafico il programma dovrà calcolare i valori che la funzione assume tra i suoi estremi, memorizzarli, e visualizzare il grafico solo dopo aver utilizzato l'artificio che permette di ottenere l'effetto desiderato. Il grafico sarà visualizzato disegnando le cosiddette «curve di livello»; sarà quindi necessaria la disponibilità di un'area di memoria dove andare a leggere i valori che serviranno a tracciare il disegno, memorizzati in precedenza. Per disegnare ogni curva ricorreremo ad una approssimazione ottenuta tramite una spezzata.

L'operazione che ci permetterà di ottenere l'effetto tridimensionale consiste nel normalizzare rispetto alla terza due delle tre coordinate che individuano un punto nello spazio. Per poter effettuare la normalizzazione, il programma calcola le coordinate del punto interessato e ne divide i valori per il valore della X. Ciò permetterà di disegnare la figura con un buon effetto prospettico, avendo ottenuto le due coordinate dei punti su un piano in funzione della terza, che è stata eliminata.

Per memorizzare i dati riguardanti i punti dovremo utilizzare una matrice, che sarà stata dimensionata in base al numero dei punti che si vogliono prendere in esame. Quindi, tanto più alto è il grado di definizione che vogliamo per la nostra figura, tanto più grande sarà la nostra matrice, con l'avvertenza però di non inserire cifre eccessivamente grandi per i punti da prendere in esame, poiché la memoria del CBM-64 non sarebbe in grado di contenere l'intera matrice.
A seconda degli intervalli di definizione assegnati alle due variabili indipendenti della funzione, è probabile che lo schermo non possa contenere l'intero grafico, oppure che lo realizzi nella sola parte centrale, non consentendone una buona visualizzazione. Per ovviare a questo inconveniente saranno calcolati appositi fattori di scala che consentiranno di disegnare il grafico sempre nella massima espansione consentita dal video.

Per ottenere un buon risultato grafico utilizzeremo anche in questa applicazione la bit-map, e per tracciare i segmenti che compongono la figura distingueremo tre possibili casi: che il segmento risulti parallelo all'asse delle ascisse, che risulti parallelo all'asse delle ordinate o che non sia parallelo né all'uno né all'altro. Ciò fatto si procederà all'accensione dei punti del segmento, come già si è visto in precedenza (pag. 96).

Diagrammi di flusso

Il programma esegue in primo luogo l'inizializzazione delle costanti del C-64 e la stampa del titolo.
Quindi visualizza sullo schermo un messaggio in cui chiede se si vuole modificare la funzione, inserita in una linea del programma. Qualora si cambi la funzione bisognerà rilanciare di nuovo l'esecuzione del programma con il comando RUN.
Si passa poi a verificare se la base del programma è stata spostata alla locazione 16384; in caso negativo si visualizzano le istruzioni necessarie per effettuare lo spostamento e si arresta l'esecuzione. Nel caso che lo spostamento sia già stato effettuato, il programma continua chiedendo il livello di definizione del disegno e i valori minimi e massimi delle due variabili indipendenti (X,Y).
Letti i dati si esegue una verifica, controllando se i massimi di X e Y sono minori o uguali ai minimi; se è così, il programma torna a visualizzare la richiesta di immissione dati.
Accertata la legalità dei dati immessi, il programma inizia a determinare i valori di X e Y dei punti da graficare; quindi carica in una matrice i valori normalizzati che determina per X e Y, utilizzando il passo di X e Y calcolato in precedenza.
Nel passaggio successivo esegue la lettura dei valori precedentemente memorizzati, ed esegue i

confronti necessari per individuare il minimo ed il massimo valore di Y e di Z. A questo punto, in base ai dati in suo possesso, il programma inizia il calcolo dei fattori di scala orizzontali e verticali, per ottenere la massima espansione della figura da rappresentare sullo schermo.
Qui inizia la fase che si riferisce al modo grafico; il programma chiama la bit-map, la «ripulisce» da eventuali disegni precedenti e la colora in blu. Espletate le operazioni di calcolo e di attivazione del modo grafico, inizia a visualizzare la figura sullo schermo, accendendo tutti i punti individuati dai valori presenti nelle colonne e nelle righe della matrice M. Eseguito il disegno, premendo un tasto qualsiasi il programma terminerà la sua esecuzione, e riporterà lo schermo nella configurazione iniziale.

Il programma

Nella linea 330 è memorizzata la funzione di due variabili (che può essere modificata) della quale si vuole tracciare il grafico tridimensionale. Si carica poi il titolo del programma nella stringa PG$ (linea 350) e si chiama la routine di visualizzazione 2990. Questa chiama a sua volta la routine 2720, che inizializza le costanti del C-64. Nella visualizzazione successiva lo sfondo è colorato in blu (linea 380) e sono presentate le informazioni riguardanti l'eventuale modifica della funzione (linee 410÷460). Per modificare la funzione si deve premere il tasto S, che visualizza la linea 330 (linea 480) in cui si dovrà scrivere l'espressione della funzione prescelta. Subito dopo è visualizzata la richiesta dell'intervallo (range) di valori della X e della Y. La parte successiva (immissione di S o C) è demandata alla routine 2290, che cancella la zona dello schermo dove sarà inserita la stringa alfanumerica; in questo caso la stringa è di un solo carattere, avendo assegnato nella linea 470 alla variabile ZL (lunghezza massima della stringa alfanumerica) il valore 1. La stessa 2290 legge il carattere; se il carattere non è stato inserito fa lampeggiare il cursore sul video (linee 2390÷2410), se invece è stato inserito memorizza il suo codice ASCII (linea 2450) e riporta la lunghezza

```
100 REM ***************************
110 REM GRAFICA TRIDIMENSIONALE
120 REM ***************************
140 REM VARIABILI UTILIZZATE
160 REM M(*,*,2):MEMORIZZA LE TRIPLE CARTESIANE DA VISUALIZZARE
170 REM I,J      :CONTATORI DI CICLO
180 REM P        :NUMERO DI PUNTI DA PRENDERE IN ESAME
190 REM P1,P2    :DIPENDONO DA 'P'
200 REM PX,PY    :PASSO PER LA X E LA Y IN MODO DA ESAMINARE SOLO 'P' PUNTI
210 REM X,Y      :COPPIE DI PUNTI PER LE QUALI SI CALCOLA IL VALORE DI 'Z'
220 REM LX,HX    :MINIMO E MASSIMO PER 'X'
230 REM LY,HY    :MINIMO E MASSIMO PER 'Y'
240 REM LZ,HZ    :MINIMO E MASSIMO PER 'Z'
250 REM SY,SZ    :FATTORI DI SCALA PER 'Y' E 'Z'
260 REM CO       :COLORE DELLA MEMORIA BIT-MAP
270 REM XP,YP    :COORDINATE DEL PUNTO DI PARTENZA DI UN SEGMENTO DA VISUALIZZARE
280 REM XS,YS    :COORDINATE DEL PUNTO DI ARRIVO
310 REM LA FUNZIONE DI DUE VARIABILI
320 REM MEMORIZZATA ALLA LINEA 330 E' QUELLA DA VISUALIZZARE
330 DEF FNZ(Z)=SIN(X+Y)
340 REM INIZIA IL PROGRAMMA
350 PG$="GRAFICA TRIDIMENSIONALE"
360 GOSUB 2990
370 REM SFONDO BLU
380 POKE VI+32,6:POKE VI+33,6
390 REM STAMPO INFORMAZIONI
410 PRINT "LA FUNZIONE DA VISUALIZZARE E' "
420 PRINT "SCRITTA ALLA LINEA 330."
430 PRINT "DIGITARE :"
440 PRINT "'S' PER MODIFICARLA;"
450 PRINT "'C' PER CONTINUARE;"
460 PRINT "[S/C]? ";
470 ZL=1:GOSUB 2290
480 IF IN$="S" THEN PRINT"":LIST 330:END
490 REM ORA CONTROLLO CHE IL PROGRAMMA SIA MEMORIZZATO DALLA LOCAZIONE 16384
500 IF PEEK(44)=64 THEN GOTO 640
510 REM AVVERTO CHE IL PROGRAMMA NON PUO' ANDARE IN ESECUZIONE
520 PRINT ""
530 PRINT "ATTENZIONE HAI DIMENTICATO"
540 PRINT "DI IMMETTERE I SEGUENTI COMANDI:"
550 PRINT "1)POKE 44,64"
560 PRINT "2)POKE 16384,0"
570 PRINT "PRIMA DI IMMETTERLI RICORDA"
580 PRINT "DI SALVARE IL PROGRAMMA, SE QUESTO"
590 PRINT "NON E' GIA' PRESENTE SU NASTRO,"
600 PRINT "E POI DI RICARICARLO":END
610 REM INIZIALIZZO COSTANTI
620 REM ORA INIZIA IL PROGRAMMA
630 REM CHIEDO IL NUMERO DI PUNTI DA BATTERE PER X ED Y
640 PRINT "";
650 PRINT "QUANTI PUNTI DEVO MARCARE ? ";
660 ZL=2:GOSUB 2290
670 P=VAL(IN$)
680 IF P>1 THEN GOTO 710
690 REM IL NUMERO DI PUNTI DA MARCARE E' TROPPO BASSO, RIPETERE INPUT
700 GOTO 640
710 P1=P-1
720 P2=P*P-1
730 REM LIMITI PER LA X
740 PRINT "VALORE MINIMO DI X (>0) ? ";
750 ZL=10:GOSUB 2290:PRINT
760 REM CONTROLLO CHE X SIA > 0
770 LX=VAL(IN$)
780 IF LX<=0 THEN GOTO 740
790 PRINT "VALORE MASSIMO DI X ? ";
800 GOSUB 2290:PRINT
810 HX=VAL(IN$)
820 REM CONTROLLO CHE IL MASSIMO PER X SIA MAGGIORE DEL MINIMO
```

della stringa nella variabile Z9 (linea 2450), che sarà utilizzata per controllare che non sia stato inserito più del massimo dei caratteri previsti (linea 2530). Inoltre, se il carattere inserito non è un carattere alfanumerico, si controlla se si tratta di un DELETE, nel qual caso si cancella sul video e nella stringa IN$ l'ultimo carattere immesso (linee 2650, 2660), oppure di un RETURN (linea 2610), e allora si torna alla linea 480.
Poiché il programma si avvale della bit-map, per poterlo caricare è necessario spostare la base del programma stesso in un indirizzo di memoria diverso da quello in cui il sistema carica abitualmente i programmi-utente, quindi sono visualizzate le istruzioni necessarie ad eseguire questa operazione se non è già stata effettuata (linee 510÷600).
È poi chiesto quanti punti occorre considerare (linea 650), valore determinante per stabilire il numero di segmenti che saranno poi visualizzati. È chiamata quindi la routine 2290 per la gestione del quadro, con la sola differenza rispetto alla chiamata precedente che la lunghezza massima della stringa alfanumerica è stabilita pari a due caratteri (linea 660). Un numero minore di due in questa fase non sarà accettato, in quanto troppo piccolo (linee 680÷700).
Alle linee 740÷830 è chiesto di inserire il valore minimo e massimo per la variabile X (linee 740, 750), e si controlla che il valore massimo non sia minore o uguale al minimo (linea

```
830 IF HX<=LX THEN GOTO 740
840 REM LEGGO MINIMO E MASSIMO PER LA Y
850 PRINT "□";
860 PRINT "VALORE MINIMO DI Y ? ";
870 GOSUB 2290:PRINT
880 LY=VAL(IN$)
890 PRINT "▒VALORE MASSIMO DI Y ? ";
900 GOSUB 2290:PRINT
910 HY=VAL(IN$)
920 REM CONTROLLO CHE IL MASSIMO PER Y SIA MAGGIORE DEL MINIMO
930 IF HY<=LY THEN GOTO 850
931 FOR A=0 TO 400:NEXT A
935 PRINT"▒▒▒▒▒▒▒▒▒▒▒ATTENDERE, PREGO■"
940 REM ORA HO I LIMITI PER X ED Y
950 REM ORA CALCOLO IL PASSO PER X ED Y
960 PX=(HX-LX)/P1
970 PY=(HY-LY)/P1
980 REM DIMENSIONO LA MATRICE DEI PUNTI DA VISUALIZZARE
990 DIM M(P1,P1,2)
1000 REM ASSEGNO E NORMALIZZO RISPETTO AD X I VALORI DELLA MATRICE M
1010 FOR I=0 TO P1
1020 FOR J=0 TO P1
1030 X=LX+PX*I
1040 Y=LY+PY*J
1050 M(I,J,1)=Y/X
1060 M(I,J,2)=FNZ(Z)/X
1070 NEXT J
1080 NEXT I
1090 REM ORA POSSO VISUALIZZARE LE COPPIE Y,Z NORMALIZZATE RISPETTO AD X
1100 REM CALCOLO MINIMO E MASSIMI PER Y E Z
1110 LY=M(0,0,1)
1120 HY=LY
1130 LZ=M(0,0,2)
1140 HZ=LZ
1150 FOR I=0 TO P1
1160 FOR J=0 TO P1
1170 IF M(I,J,1)<LY THEN LY=M(I,J,1)
1180 IF M(I,J,1)>HY THEN HY=M(I,J,1)
1190 IF M(I,J,2)<LZ THEN LZ=M(I,J,2)
1200 IF M(I,J,2)>HZ THEN HZ=M(I,J,2)
1210 NEXT J
1220 NEXT I
1230 REM ORA CALCOLO I FATTORI DI SCALA PER Y E Z
1240 SY=319/(HY-LY)
1250 SZ=199/(HZ-LZ)
1260 REM VISUALIZZO LA FUNZIONE
1270 REM INIZIALIZZO IL MODO GRAFICO
1280 GOSUB 1670
1290 REM CANCELLO LO SCHERMO
1300 GOSUB 1760
1310 REM COLORO LO SFONDO IN BLU
1320 CO=6:GOSUB 1790
1330 REM POSSO DISEGNARE
1340 FOR I=0 TO P1
1350 XP=(M(I,0,1)-LY)*SY
1360 YP=(M(I,0,2)-LZ)*SZ
1370 FOR J=1 TO P1
1380 XS=(M(I,J,1)-LY)*SY
1390 YS=(M(I,J,2)-LZ)*SZ
1400 GOSUB 1870
1410 XP=XS:YP=YS
1420 NEXT J
1430 NEXT I
1440 REM LO STESSO PER Z.
1450 FOR I=0 TO P1
1460 XP=(M(0,I,1)-LY)*SY
1470 YP=(M(0,I,2)-LZ)*SZ
1480 FOR J=1 TO P1
```

830); se è così i valori inseriti non sono considerati, e riappare la richiesta di immissione. La gestione della routine 2290 stabilisce in dieci caratteri la lunghezza massima della stringa alfanumerica.
Le linee 840÷930 ripetono anche sulla variabile Y la stessa operazione fatta sulla X.
Alle linee 960, 970 è stabilito il passo per X e Y, che dipende dai valori massimi e minimi inseriti per le due incognite rapportate al numero dei punti da visualizzare.
Di seguito si dimensiona la matrice M (linea 990), vi si memorizzano i dati dei punti da visualizzare e si normalizza rispetto a X (linee 1010÷1080), cioè si divide il valore di ogni Y per X, quindi dato che ogni X divisa per se stessa dà come risultato 1, partendo dai precedenti valori si calcolano i valori sull'asse Z [FNZ(Z)/X]. In questo modo rimangono solo l'asse Y (orizzontale) e Z (verticale).
Alle linee 1110÷1220 il programma calcola i valori minimi e massimi per Y e Z, assumendo inizialmente il primo valore di Y trovato nella matrice sia come massimo sia come minimo, e confrontando poi ogni Y della matrice con HY e LY, in modo da cambiare il valore di HY se il valore di Y risultasse maggiore, o quello di LY se il valore delle Y risultasse minore. Dopo aver letto tutta la matrice si troverà in HY il massimo valore di Y e in LY il minimo. Nella stessa maniera si troveranno il minimo e il massimo per Z. È ora necessario calcolare i fattori di scala per Y e Z, per poter ridurre il disegno a dimensioni tali da

```
1490 XS=(M(J,I,1)-LY)*SY
1500 YS=(M(J,I,2)-LZ)*SZ
1510 GOSUB 1870
1520 XP=XS:YP=YS
1530 NEXT J
1540 NEXT I
1550 GET A$:IF A$="" THEN GOTO 1550
1560 REM RITORNO AL MODO CARATTERE
1570 GOSUB 2170
1580 REM IL PROGRAMMA E' TERMINATO
1590 PRINT "[illegible]"
1600 END
1610 REM **********************************************************************
1620 REM I SOTTOPROGRAMMI PER LA GESTIONE DELLA GRAFICA, SONO SCRITTI
1630 REM IN MODO DA RENDERE IL PIU' VELOCE POSSIBILE LA LORO AZIONE
1640 REM **********************************************************************
1650 REM ROUTINE:INIZIALIZZA IL MODO VIDEO BIT-MAP E LE VARIABILI INTERESSATE
1660 REM ATTIVO IL MODO BIT-MAP
1670 POKE VI+17,PEEK(VI+17) OR 32
1680 REM LA MEMORIA BIT-MAP E' ALLA LOCAZIONE 8192
1690 POKE VI+24,PEEK(VI+24) OR 8
1700 BM=8192:Z0=8:Z1=320:Z2=7:Z3=1/Z0
1710 FOR I=0 TO 7
1720 Z3(I)=2↑I
1730 NEXT I
1740 RETURN
1750 REM ROUTINE:CANCELLO LA MEMORIA BIT-MAP
1760 Z9=0:FORI=BMTOBM+7999:POKEI,Z9:NEXT
1770 RETURN
1780 REM ROUTINE:IMPOSTO LO SFONDO NEL COLORE CO
1790 FORI=1024TO2023:POKEI,CO:NEXT
1800 RETURN
1810 REM ROUTINE:ACCENDE IL PUNTO DI COORDINATE XC,YC
1820 BY=BM+INT(YC*Z3)*Z1+INT(XC*Z3)*Z0+(YCANDZ2)
1830 POKEBY,PEEK(BY)ORZ3(Z2-(XCANDZ2))
1840 RETURN
1850 REM ROUTINE:TRACCIA UNA RETTA DA XP,YP A XS,YS
1860 REM LA RETTA E' PARALLELA ALL'ASSE XC?
1870 IF YP=YS THEN GOTO 2000
1880 REM LA RETTA E' PARALLELA ALL'ASSE YC?
1890 IF XS=XP THEN GOTO 2050
1900 REM POSSO CALCOLARE IL COEFFICIENTE ANGOLARE
1910 M=(YP-YS)/(XP-XS)
1920 IF ABS(M)>1 THEN M=1/M:GOTO 2100
1930 REM CALCOLO INTERCETTA ALL'ORIGINE PER UNA VARIAZIONE DELLE XC
1940 Q=YS-XS*M
1950 REM TRACCIO LA RETTA
1960 FOR XC=XP TO XS STEP SGN(XS-XP)
1970 YC=M*XC+Q:GOSUB1820:NEXT
1980 RETURN
1990 REM LA RETTA E' PARALLELA ALL'ASSE XC
2000 YC=YP
2010 FOR XC=XP TO XS STEP SGN(XS-XP)
2020 GOSUB1820:NEXT
2030 RETURN
2040 REM LA RETTA E' PARALLELA ALL'ASSE YC
2050 XC=XP
2060 FOR YC=YP TO YS STEP SGN(YS-YP)
2070 GOSUB1820:NEXT
2080 RETURN
2090 REM CALCOLO INTERCETTA ALL'ORIGINE PER UNA VARIAZIONE DELLE YC
2100 Q=XS-YS*M
2110 REM TRACCIO LA RETTA
2120 FOR YC=YP TO YS STEP SGN(YS-YP)
2130 XC=M*YC+Q:GOSUB1820:NEXT
2140 RETURN
2150 REM ROUTINE:TERMINA IL MODO VIDEO BIT-MAP
2160 REM SPENGO IL MODO BIT-MAP
```

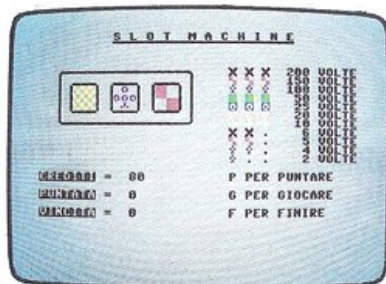
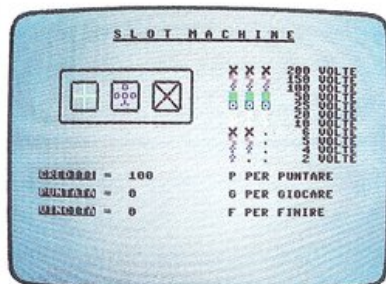
permettere la visualizzazione (linee 1240, 1250).
Alle linee 1280÷1320 si inizializza il modo grafico utilizzando la routine 1670, che attiva la bit-map e la colloca in una determinata area di memoria (linee 1690, 1700), cancella ciò che vi era stato precedentemente memorizzato (linea 1760) e colora lo sfondo in blu (linea 1790).
Prima di poter disegnare la figura si devono moltiplicare per i fattori di scala precedentemente trovati tutte le coordinate dei punti da disegnare (linee 1340÷1390). La routine 1870 verifica se il segmento da tracciare è parallelo all'asse delle X o all'asse delle Y, e ne calcola l'inclinazione se non è parallelo a nessuno dei due assi (linea 1910). Di seguito traccia i segmenti di retta a seconda dei tre casi (linee 2000, 2050, 2100) utilizzando la routine 1820, che accende punto dopo punto il segmento sul video.
La linea 1410 assume come coordinate di partenza del successivo segmento quelle del punto di arrivo del segmento appena tracciato e ripete il ciclo finché non avrà disegnato tutti i punti calcolati per l'asse Y.
Alle linee 1400÷1540 si determinano e si moltiplicano per i fattori di scala precedentemente trovati tutte le coordinate dei punti da tracciare sull'asse Z, utilizzando un procedimento analogo a quello usato per l'asse Y.
Per finire la linea 1550 attende che sia digitato un qualsiasi tasto per spegnere il modo grafico (linea 1570) e terminare l'esecuzione.

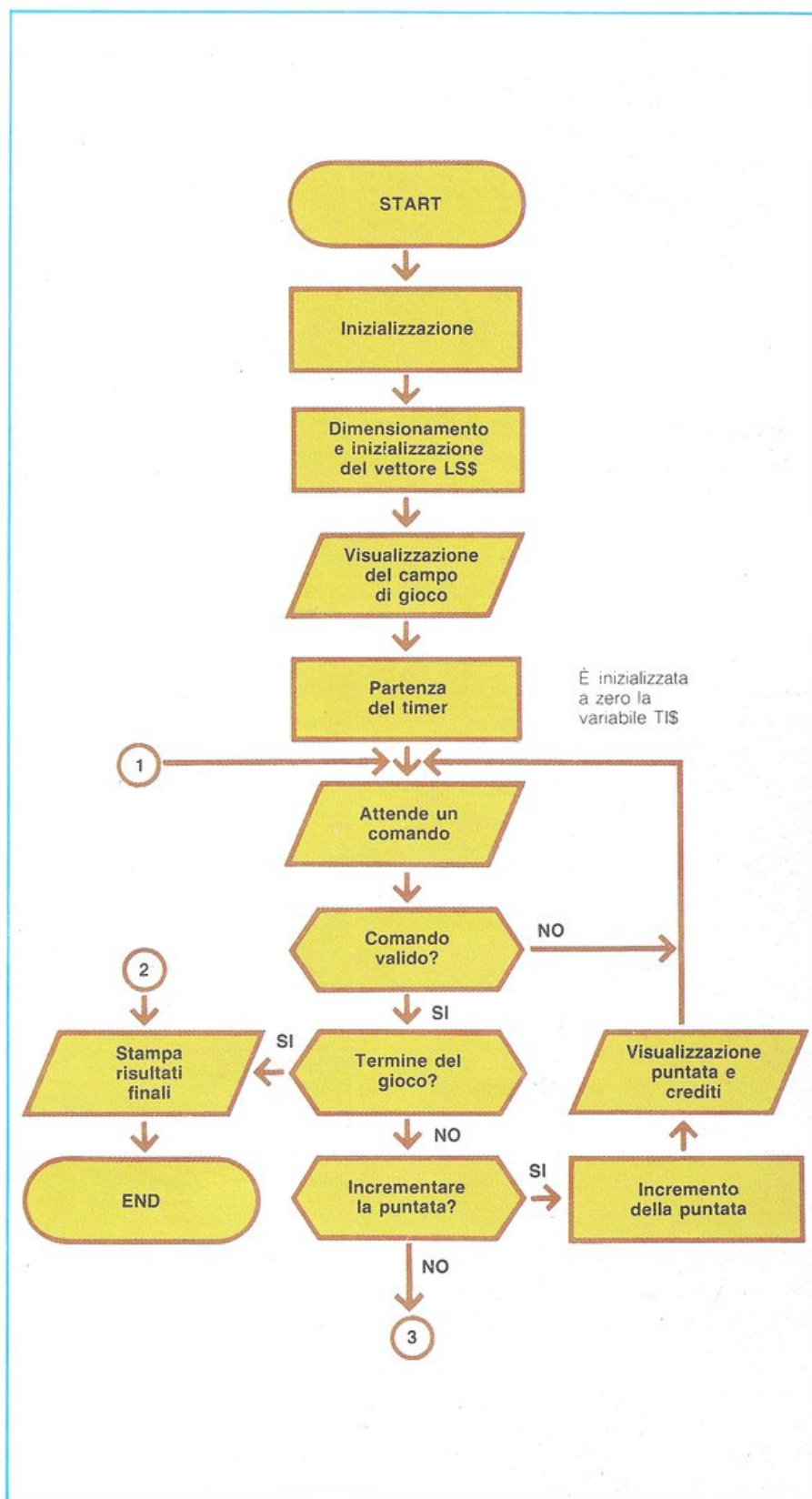
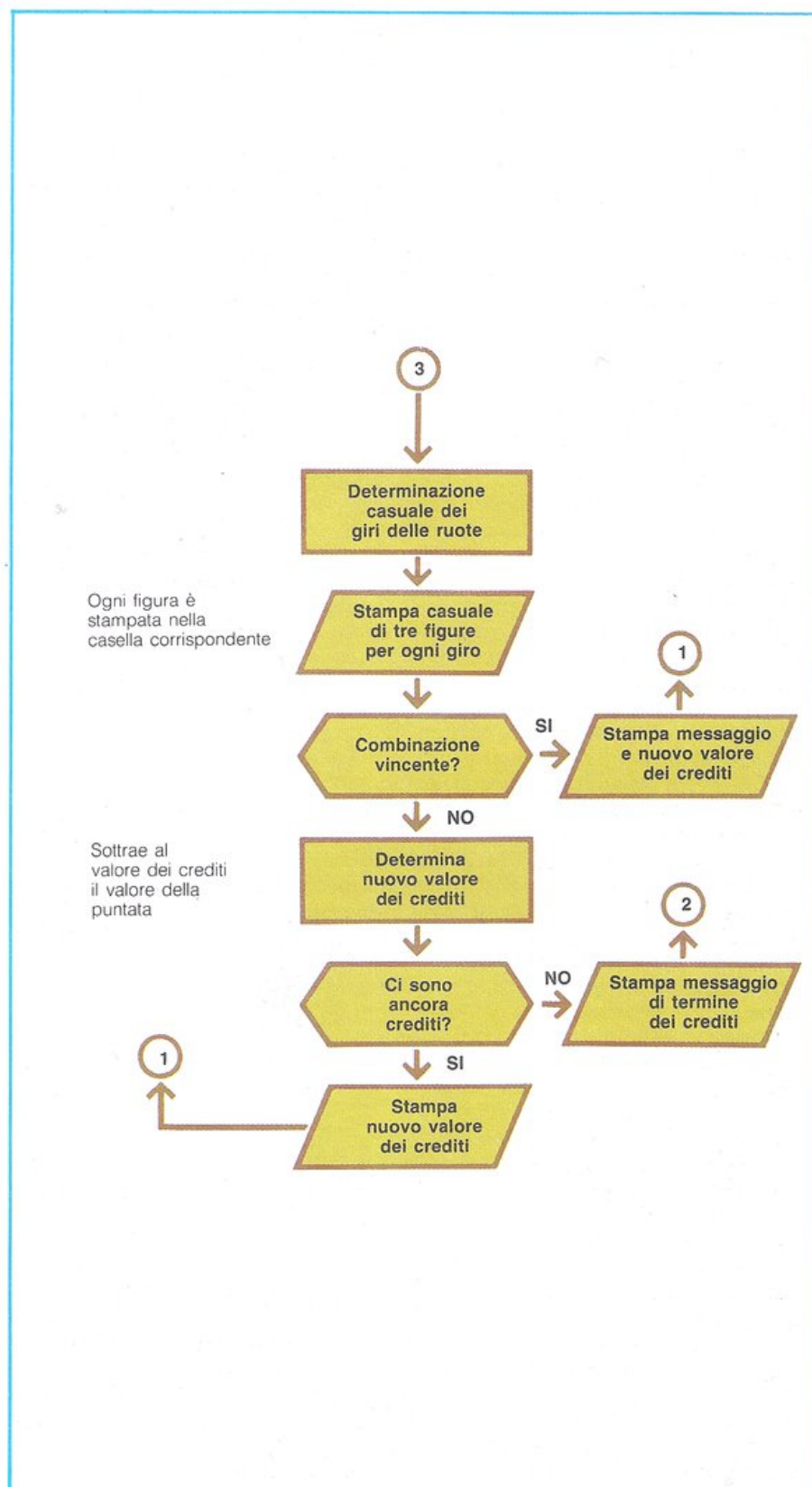
```
2170 POKE VI+17,PEEK(VI+17) AND 223
2180 POKE VI+24,PEEK(VI+24) AND 247
2190 RETURN
2210 REM ROUTINE:GESTISCE L'INPUT DI UNA STRINGA ALFANUMERICA
2220 REM LE VARIABILI UTILIZZATE SONO:Z6,Z7,Z8,Z9,Z8$,IN$
2230 REM IN INGRESSO VUOLE IL VALORE ZL CHE E' LA LUNGHEZZA MASSIMA
2240 REM DELLA STRINGA DA LEGGERE
2250 REM IN USCITA DA' LA STRINGA LETTA IN IN$ E LA SUA LUNGHEZZA IN Z9
2270 REM CANCELLO LA ZONA DI SCHERMO IN CUI VERRA' DIGITATA LA STRINGA
2290 FOR Z8=1 TO ZL: PRINT " ";:NEXT Z8
2300 FOR Z8=1 TO ZL: PRINT "▮";:NEXT Z8
2310 IN$="":Z7=TI
2330 REM LEGGO UN CARATTERE
2350 GET Z8$:IF Z8$<>"" THEN 2450
2370 REM ACCENSIONE E SPEGNIMENTO DEL CURSORE
2390 IF Z7<TI AND NOT(Z6) THEN PRINT "▒▮";:Z6=NOT(Z6):Z7=TI+15
2400 IF Z7<TI AND Z6 THEN PRINT " ▮";:Z6=NOT(Z6):Z7=TI+15
2410 GOTO 2350
2430 REM E' STATO DIGITATO UN CARATTERE
2450 Z8=ASC(Z8$):Z9=LEN(IN$)
2470 REM SE NON E' UN CARATTERE ALFANUMERICO, DEVE ESSERE UN RETURN O DELETE
2490 IF NOT(Z8>31 AND Z8<96) THEN GOTO 2610
2510 REM CONTROLLO CHE NON SIA STATA SUPERATA LA LUNGHEZZA MASSIMA
2530 IF Z9=ZL THEN GOTO 2350
2550 REM LO AGGIUNGO ALLA STRINGA IN$
2570 IN$=IN$+Z8$:PRINT Z8$;:GOTO 2350
2590 REM SE E' UN RETURN, HO TERMINATO LA LETTURA
2610 IF Z8=13 THEN PRINT " ▮";:RETURN
2630 REM SE E' DELETE, CANCELLO IN IN$ E SUL VIDEO L' ULTIMO CARATTERE DIGITATO
2650 IF Z8=20 AND Z9>0 THEN IN$=LEFT$(IN$,Z9-1):PRINT " ▮▮";:GOTO 2350
2660 GOTO 2350
2680 REM ROUTINE:INIZIALIZZAZIONE COSTANTI
2700 REM CIRCUITO VIDEO
2720 VI=53248
2740 REM CIRCUITO SUONO
2760 SI=54272
2780 REM MEMORIA VIDEO
2800 MV=1024
2820 REM MEMORIA COLORE
2840 MC=55296
2860 REM COSTANTI DI USO COMUNE
2880 ZL=9
2900 REM INIZIALIZZAZIONE CHIP SUONO
2920 FOR I=0 TO 24
2930 POKE SI+I,0
2940 NEXT I
2950 RETURN
2970 REM ROUTINE:STAMPA IL TITOLO DEL PROGRAMMA MEMORIZZATO IN PG$
2990 GOSUB 2720
3000 POKE VI+32,15
3010 POKE VI+33,15
3020 PRINT "▒▒▒▒▒▒";
3030 PRINT TAB(6);"┌──────────────────────────┐"
3040 FOR I=1 TO 5
3050 PRINT TAB(6);"|                          |"
3060 NEXT I
3070 PRINT TAB(6);"└──────────────────────────┘"
3080 PRINT TAB(6);"▒▒▒▒▒▒▒▒DIGITA ▒RETURN▒ PER PROSEGUIRE"
3090 PRINT "▒▒▒▒▒▒"
3100 FOR I=1 TO 5
3110 PRINT TAB(7);
3120 FOR J=1 TO 26
3130 PRINT "▒▒ ";
3140 NEXT J
3150 PRINT
3160 NEXT I
3180 REM ORA SCRIVO IL TITOLO
3200 PRINT "▒▒▒▒▒▒▒▒▒▒";TAB((40-LEN(PG$))/2);"▒▒";PG$
3210 GET Z9$
3220 IF Z9$<>CHR$(13) THEN GOTO 3210
3230 PRINT "▒▒";
3240 RETURN
```

Slot machine

Il programma che presentiamo adesso ci permetterà di simulare il funzionamento di una slot machine.
I componenti essenziali di una slot machine sono le tre ruote, su ognuna delle quali sono disegnate ad intervalli regolari alcune figure, ed un comando, generalmente una leva, che aziona alcuni meccanismi i quali imprimono alle ruote un movimento. Quando le ruote si fermano, ognuna di esse presenta al giocatore una figura. Il gioco consiste nell'effettuare una puntata, azionare il comando che permette di far girare le ruote, ed attendere: a seconda della combinazione che le tre figure avranno assunto sarà assegnata una determinata vincita.

Analisi del problema

Il nostro progetto permetterà di usare il calcolatore come se fosse una slot machine, assegnando al giocatore un certo numero di crediti e prevedendo la possibilità di effettuare una puntata, di far ruotare le figure e di interrompere il gioco quando si vuole. Saranno visualizzate le combinazioni vincenti delle figure con le relative vincite, le istruzioni per giocare, le informazioni relative ai crediti disponibili, alla puntata effettuata e alla vincita eventualmente realizzata.
Dopo l'inizializzazione delle costanti, si dovranno scegliere e memorizzare le figure utilizzate nel gioco. Il numero, il colore e l'aspetto delle figure saranno memorizzati in un solo array (SL$). Visualizzato successivamente il campo di gioco, sarà necessario inizializzare le variabili; in particolare, si memorizzerà un valore positivo (100 nel nostro caso) nella variabile CR che controlla i crediti disponibili, e zero nelle altre (puntata e vincita). Il programma dovrà quindi prelevare un carattere dalla tastiera, che sarà P per effettuare la puntata, G per giocare ed F per finire il gioco. Qualora il giocatore desideri puntare, il programma dovrà controllare se vi sono ancora crediti disponibili, incrementare la puntata e decrementare i crediti; qualora invece si desideri finire il gioco, sarà visualizzato il tempo trascorso dall'inizio delle giocate.
Se infine si imposta la richiesta di giocare, il programma, dopo aver controllato se è stata precedentemente effettuata la puntata, dovrà attivare l'effetto sonoro e far ruotare le figure. Il numero dei giri che dovrà fare ciascuna ruota sarà determinato in modo casuale mediante l'uso della funzione RND, e varierà tra 50 e 100 per la terza ruota; la seconda ruota effettuerà un numero di giri doppio rispetto alla terza e la prima triplo. Per simulare il movimento delle ruote di una vera slot machine useremo un ciclo, controllato da una variabile che fornisce il numero dei giri da effettuare. Visualizzeremo così una veloce successione arbitraria di figure, scelte di volta in volta utilizzando la funzione RND. Il valore fornito da tale funzione funge da indice per accedere all'array delle figure (SL$) e per determinare quindi la figura da visualizzare. Il programma dovrà poi determinare se si è verificata una combinazione vincente (useremo a questo scopo istruzioni condizionate IF... THEN...) e, in caso affermativo, calcolare l'entità della vincita moltiplicando la puntata per il coefficiente relativo alla combinazione verificatasi. Il computer emetterà un fischio prolungato, mentre il programma sommerà l'importo della vincita ai crediti disponibili e azzererà la puntata per predisporre una nuova partita. Il programma terminerà quando non vi saranno più crediti, o in seguito ad una scelta del giocatore. Per ottenere la visualizzazione della durata del gioco si dovrà inizializzare la variabile di sistema TI$ all'inizio del gioco. Al termine sarà sufficiente visualizzare il valore di TI$ utilizzando la funzione MID$ per evidenziare separatamente le ore, i minuti e i secondi trascorsi.

Diagrammi di flusso

Dopo aver stampato il titolo e inizializzato le costanti, il programma passa a dimensionare e a inizializzare, attraverso le istruzioni DATA, il vettore LS$. Questo vettore contiene i caratteri delle figure che ruoteranno. Quindi si passa a presentare un'immagine video che mostra il campo di gioco, la lista comandi, i coefficienti di vincita e il valore iniziale delle variabili crediti, puntata e vincita. Nel blocco successivo avviene l'inizializzazione del timer, attraverso l'attribuzione alla variabile TI$ di una stringa di sei zeri; il contenuto di TI$ sarà aggiornato ad ogni secondo dal C-64 e ciò permetterà in qualunque momento di avere l'indicazione del tempo di gioco utilizzato. Si attende poi che il giocatore immetta un comando da tastiera scegliendolo fra quelli previsti. Qualora sia premuto il tasto F (comando di fine gioco), il programma visualizza il tempo trascorso dall'inizio del gioco e i risultati ottenuti. Se si tratta invece del comando di incremento del valore di puntata (P), questo è aumentato di un'unità, e il programma ritorna nella fase di attesa di un comando. Infine, se il comando indica di giocare (G) è calcolato in modo casuale il numero di giri delle tre ruote; la rotazione avviene attraverso un ciclo di lettura e stampa del vettore

LS$. La casualità di lettura si ottiene utilizzando la funzione RND, che genera un numero casuale compreso fra zero ed uno, mentre per far terminare la rotazione delle figure nelle caselle in tempi differiti si attribuisce ad ognuna di esse un numero di giri diverso. Terminata la fase di rotazione, si prende in considerazione la combinazione ottenuta verificando se risulta vincente o no. Se sì il programma calcola nel successivo blocco la vincita (cioè moltiplica la puntata per i coefficienti associati al valore della combinazione), quindi visualizza un messaggio di congratulazioni e aggiorna i valori delle variabili. Nel caso in cui la combinazione non sia vincente, si determina semplicemente il valore dei crediti e si verifica se detto valore è maggiore o uguale a zero. Se è maggiore di zero il programma torna ad attendere un comando per una nuova partita, sostituendo alle variabili iniziali i nuovi valori ottenuti; se è minore o uguale a zero invia un messaggio di fine esecuzione e dà il valore del tempo trascorso dall'inizio del gioco.

Il programma

Il programma comincia alla linea 260 e prosegue chiamando alla linea seguente la routine 3030, che permette la visualizzazione del titolo (in bianco su sfondo rosso) e l'inizializzazione delle costanti (linee 2760÷2990). È poi dimensionato l'array SL$ e vi si memorizzano le figure utilizzate nel gioco (linee 340÷400). Nelle linee successive è ripresentato il titolo, ed è visualizzata una tabella che fornisce le combinazioni vincenti delle figure con i rispettivi coefficienti di vincita. Si visualizza anche una griglia che contiene tre riquadri (rappresentanti le caselle della slot machine), si stampano le diciture CREDITI, PUNTATA e VINCITA (in grigio su sfondo nero), e sono presentate le istruzioni per il giocatore (linee 450÷800). Subito dopo sono inizializzate le variabili relative ai crediti (linea 820), cui è assegnato il valore 100, alla puntata (linea 830) e alla vincita (linea 840); quindi sono chiamate le routines 1740, 1800 e 1850, che ne visualizzano i valori posizionandoli alla riga desiderata. Il programma affida poi il controllo alle routines 2100, 2150 e 2200, che visualizzano nell'apposita griglia tre figure estratte in modo casuale nell'array SL$. Si passa poi ad inizializzare TI$ (tempo di gioco), ci si pone in attesa di un comando da tastiera (linee 970, 980) e ricevutolo, si

```
100 REM ***************
110 REM *SLOT MACHINE*
120 REM ***************
140 REM VARIABILI UTILIZZATE
160 REM I,J       :CONTATORI DI CICLO
170 REM CR        :MEMORIZZA I CREDITI A DISPOSIZIONE DEL GIOCATORE
180 REM PU        :VALORE DELLA PUNTATA FATTA
190 REM VC        :VALORE DELLA VINCITA CONSEGUITA
200 REM F1,F2,F3:NUMERO DELLA FIGURA VISUALIZZATA ALLA POSIZIONE 1,2,3
210 REM A         :CONTIENE IL COMANDO IMMESSO DA TASTIERA
220 REM GI        :NUMERO DI GIRI FITTIZI CHE LA PRIMA FIGURA DEVE COMPIERE
230 REM RS        :PUNTEGGIO OTTENUTO IN BASE ALLA COMBINAZIONE DELLE TRE FIGURE
231 REM SL$(7)   :CONTIENE I DISEGNI DELLE FIGURE DA VISUALIZZARE
250 REM STAMPO TITOLO ED INIZIALIZZO COSTANTI C64
260 PG$="SLOT MACHINE"
270 GOSUB 3030
280 REM DIMENSIONO ARRAY
290 DIM SL$(7)
330 REM IMMETTO NEL VETTORE SL$(7), LE FIGURE DA VISUALIZZARE
340 SL$(1)="▒\ /▒▒▒▒ X ▒▒▒▒/ \▒"
350 SL$(2)="▒▒ ▒ ▒▒▒▒▒▒▒▒▒▒▒▒ ▒ ▒"
360 SL$(3)="▒ O ▒▒▒▒OOO▒▒▒▒ ┴ ▒"
370 SL$(4)="▒▒   ▒▒▒▒   ▒▒▒▒   ▒▒"
380 SL$(5)="▒▒╭─╮▒▒▒▒|●|▒▒▒▒╰─╯▒▒"
390 SL$(6)="▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
400 SL$(7)="▒ | ▒▒▒▒├┼┤▒▒▒▒ | ▒"
440 REM ORA VISUALIZZO MASCHERA VIDEO
450 PRINT "▒▒";TAB(9);"S L O T  M A C H I N E"
460 PRINT TAB(9);"──────────────────────▒▒"
480 PRINT TAB(23);"▒X X X  ▒200 VOLTE"
490 PRINT TAB(23);"▒▒ ▒ ▒  ▒150 VOLTE"
500 PRINT TAB(23);"▒♠ ♠ ♠  ▒100 VOLTE"
510 PRINT TAB(23);"▒▒ ▒ ▒ ▒ ▒ ▒  ▒ 50 VOLTE"
520 PRINT TAB(23);"▒▒O▒ ▒O▒ ▒O▒  ▒ 25 VOLTE"
530 PRINT TAB(23);"▒▒ ▒ ▒  ▒ 20 VOLTE"
540 PRINT TAB(23);"▒+ + +  ▒ 10 VOLTE"
550 PRINT TAB(23);"▒X X ▒.    6 VOLTE"
560 PRINT TAB(23);"▒▒ ▒ ▒.    5 VOLTE"
570 PRINT TAB(23);"▒♠ ♠ ▒.    4 VOLTE"
580 PRINT TAB(23);"▒♠ ▒. .    2 VOLTE"
600 PRINT "▒▒▒▒▒";
610 PRINT "  ╭──────────────╮"
620 PRINT "  | ───  ───  ─── |"
630 PRINT "  ||   ||   ||   ||"
640 PRINT "  ||   ||   ||   ||"
650 PRINT "  ||   ||   ||   ||"
660 PRINT "  | ───  ───  ─── |"
670 PRINT "  ╰──────────────╯"
700 PRINT "▒"
710 FOR I=1 TO 15:PRINT:NEXT I
720 PRINT "▒CREDITI▒ = "
730 PRINT "▒▒PUNTATA▒ = "
740 PRINT "▒▒VINCITA▒ = "
760 PRINT "▒"
770 FOR I=1 TO 15:PRINT:NEXT I
780 PRINT SPC(23);"P PER PUNTARE"
790 PRINT "▒";SPC(23);"G PER GIOCARE"
800 PRINT "▒";SPC(23);"F PER FINIRE"
810 REM INIZIALIZZO CREDITI,PUNTATA E VINCITA
820 CR=100
830 PU=0
840 VC=0
860 REM VISUALIZZO CREDITI,PUNTATA E VINCITA
870 GOSUB 1740
880 GOSUB 1800
890 GOSUB 1850
900 REM VISUALIZZO TRE FIGURE
910 GOSUB 2100
920 GOSUB 2150
930 GOSUB 2200
940 REM INIZIALIZZO IL TEMPO DI GIOCO
950 TI$="000000"
960 REM ATTENDO UN COMANDO
970 A=PEEK(197)
980 IF A=64 THEN GOTO 970
990 IF (A<>41) AND (A<>26) AND (A<>21) THEN GOTO 970
1000 REM IL COMANDO DATO E' ACCETTABILE. LO ESEGUO
1010 IF A=41 THEN GOSUB 1900:GOTO 970
1020 IF A=21 THEN PRINT "▒▒▒BRAVO, HAI TERMINATO IN ATTIVO.":GOTO 1560
1030 REM IL COMANDO INDICA DI GIOCARE
1040 REM SE PU=0 NON SI GIOCA
1050 IF PU=0 THEN GOTO 970
1090 REM CALCOLO IL NUMERO DI GIRI DELLA TERZA FIGURA
1120 GI=INT(RND(0)*51)+50
1130 REM EMETTO UN SUONO SIMILE AD UNA RUOTA CHE GIRA
1140 POKE SI+24,15
1150 POKE SI+1,100:POKE SI,0
1160 POKE SI+5,15:POKE SI+6,255
1170 POKE SI+14,150:POKE SI+15,0
1180 POKE SI+19,15:POKE SI+20,255
```

verifica se è un comando valido (linea 990). Se il comando è P (puntata) si chiama la routine 1900, che se vi sono crediti disponibili li decrementa e incrementa la puntata. Se il comando è F (finire) è visualizzato il messaggio della linea 1020 e termina l'esecuzione del programma. Se infine il comando è G (giocata) si controlla se è stata effettuata la puntata (linea 1050) e se ciò è stato fatto si stabilisce casualmente, in un valore compreso tra 50 e 100, il numero di giri che dovrà compiere la terza ruota (linea 1120); la seconda e la prima ne compiranno rispettivamente un numero doppio e triplo. Si emette quindi un effetto sonoro che ricorda il suono di una ruota che gira (linee 1140÷1200) e vengono fatte ruotare le tre figure. Il ciclo chiama ripetutamente le routines 2200, 2150 e 2100, e visualizza in rapida successione nella griglia un gran numero di figure (linee 1220÷1260). Il programma, finito il ciclo e visualizzate quindi stabilmente tre figure, interrompe l'effetto sonoro (linee 1280÷1300) e verifica se la combinazione assunta dalle tre figure è tra quelle vincenti (linee 1320÷1450). In caso affermativo è chiamata la routine 2260, che calcola l'importo della vincita, emette un fischio prolungato (linee 2300÷2440) ed incrementa i crediti di un valore pari a quello della vincita stessa (linee 2530÷2550), accompagnando acusticamente l'operazione (linee 2570÷2620). Se non c'è stata vincita, è azzerata la puntata (linea 1460) e si controlla se vi sono ancora

```
1190 POKE SI+18,17
1200 POKE SI+4,21
1210 REM ORA FACCIO RUOTARE LE TRE FIGURE
1220 FOR I=1 TO 4*GI
1230 IF I<=GI THEN GOSUB 2200
1240 IF I<=2*GI THEN GOSUB 2150
1250 GOSUB 2100
1260 NEXT I
1270 REM ORA SPENGO IL GENERATORE DI SUONI
1280 FOR I=0 TO 24
1290 POKE SI+I,0
1300 NEXT I
1310 REM CALCOLO IL PUNTEGGIO OTTENUTO
1320 RS=0
1330 IF F1=1 AND F2=1 AND F3=1 THEN RS=200
1340 IF F1=2 AND F2=2 AND F3=2 THEN RS=150
1350 IF F1=3 AND F2=3 AND F3=3 THEN RS=100
1360 IF F1=4 AND F2=4 AND F3=4 THEN RS=50
1370 IF F1=5 AND F2=5 AND F3=5 THEN RS=25
1380 IF F1=6 AND F2=6 AND F3=6 THEN RS=20
1390 IF F1=7 AND F2=7 AND F3=7 THEN RS=10
1400 IF F1=1 AND F2=1 AND F3<>1 THEN RS=6
1410 IF F1=2 AND F2=2 AND F3<>2 THEN RS=5
1420 IF F1=3 AND F2=3 AND F3<>3 THEN RS=4
1430 IF F1=3 AND F2<>3 THEN RS=2
1440 REM CONTROLLO SE CI SONO VINCITE
1450 IF RS<>0 THEN GOSUB 2260
1460 PU=0
1470 REM VISUALIZZO PUNTATA
1480 GOSUB 1800
1490 REM CONTROLLO CHE CI SIANO ANCORA CREDITI
1510 IF CR>0 THEN GOTO 970
1520 REM I CREDITI SONO TERMINATI; VISUALIZZO TERMINE GIOCO
1540 PRINT "⌝"
1550 PRINT "▓▓▓▓▓HAI TERMINATO I CREDITI."
1560 PRINT "▓HAI GIOCATO PER"
1570 PRINT "▓";MID$(TI$,1,2);"......OR";
1580 IF MID$(TI$,1,2)="01" THEN PRINT "A"
1590 IF MID$(TI$,1,2)<>"01" THEN PRINT "E"
1600 PRINT "▓";MID$(TI$,3,2);"...MINUT";
1610 IF MID$(TI$,3,2)="01" THEN PRINT "O"
1620 IF MID$(TI$,3,2)<>"01" THEN PRINT "I"
1630 PRINT "▓";MID$(TI$,5,2);"..SECOND";
1640 IF MID$(TI$,5,2)="01" THEN PRINT "O"
1650 IF MID$(TI$,5,2)<>"01" THEN PRINT "I"
1660 REM PULISCO IL BUFFER DI INGRESSO
1670 FOR I=1 TO 10
1680 GET A$
1690 NEXT I
1700 PRINT "▓▓▓▓"
1710 END
1720 REM INIZIANO LE ROUTINES
1730 REM ROUTINE:VISUALIZZA I CREDITI
1740 POKE 214,15:PRINT
1760 PRINT SPC(10);"      "
1770 PRINT "⌝";SPC(10);CR
1780 RETURN
1790 REM ROUTINE:VISUALIZZA PUNTATA
1800 POKE 214,17:PRINT
1810 PRINT SPC(10);"      "
1820 PRINT "⌝";SPC(10);PU
1830 RETURN
1840 REM ROUTINE:VISUALIZZA VINCITA
1850 POKE 214,19:PRINT
1860 PRINT SPC(10);"      "
1870 PRINT "⌝";SPC(10);VC
1880 RETURN
1890 REM ROUTINE:INCREMENTA PUNTATA, SE POSSIBILE
1900 IF CR=0 THEN RETURN
1910 CR=CR-1
1920 PU=PU+1
1930 REM VISUALIZZO CREDITI
1940 GOSUB 1740
1950 REM VISUALIZZO PUNTATA
1960 GOSUB 1800
1970 REM EMETTO UN BEEP
1980 POKE SI,0
1990 POKE SI+1,50
2000 POKE SI+24,15
2010 POKE SI+5,15
2020 POKE SI+6,0
2030 POKE SI+4,17
2050 FOR I=1 TO 50:NEXT I
2060 POKE SI+24,0
2070 POKE SI+4,0
2080 RETURN
2090 REM ROUTINE:VISUALIZZA LA FIGURA 1
2100 POKE 214,5:PRINT
2110 F1=INT(RND(0)*7)+1
2120 PRINT SPC(4);SL$(F1)
```

crediti disponibili (linea 1510). Nel caso che i crediti siano finiti è visualizzato il messaggio della linea 1550, assieme alla durata della giocata, altrimenti il programma si pone di nuovo in attesa di un comando.

```
2130 RETURN
2140 REM ROUTINE:VISUALIZZA LA FIGURA 2
2150 POKE 214,5:PRINT
2160 F2=INT(RND(0)*7)+1
2170 PRINT SPC(9);SL$(F2)
2180 RETURN
2190 REM ROUTINE:VISUALIZZA LA FIGURA 3
2200 POKE 214,5:PRINT
2210 F3=INT(RND(0)*7)+1
2220 PRINT SPC(14);SL$(F3)
2230 RETURN
2240 REM ROUTINE:GESTIONE VINCITA
2260 VC=PU*RS:REM CALCOLO IL NUMERO DI CREDITI VINTI
2280 GOSUB 1850:REM VISUALIZZO LA VINCITA
2290 REM EMETTO UN FISCHIO PROLUNGATO
2300 POKE SI+24,15
2310 POKE SI+5,15+2*16
2320 POKE SI+6,15*16
2330 POKE SI,0
2340 POKE SI+4,17
2350 FOR J=1 TO 8
2360 FOR I=1 TO 255 STEP 16
2370 POKE SI+1,I
2380 NEXT I
2390 FOR I=255 TO 1 STEP -16
2400 POKE SI+1,I
2410 NEXT I
2420 NEXT J
2430 POKE SI+4,0
2440 POKE SI+24,0
2450 REM ORA INCREMENTO I CREDITI, DECREMENTO LA VINCITA, EMETTENDO UN BEEP
2460 POKE SI+24,15
2470 POKE SI,0
2480 POKE SI+1,0
2490 POKE SI+4,33
2500 POKE SI+5,15
2510 POKE SI+6,240
2530 VC=VC-1
2550 CR=CR+1
2560 REM VISUALIZZO LE DUE COSE
2570 GOSUB 1740
2580 GOSUB 1850
2590 REM EMETTO UN BEEP
2600 POKE SI+1,90
2610 FOR I=1 TO 5:NEXT I
2620 POKE SI+1,0
2630 REM CONTROLLO SE HO TERMINATO
2640 IF VC>0 THEN GOTO 2530
2650 REM SPENGO IL GENERATORE DI SUONI
2660 FOR I=0 TO 24
2670 POKE SI+I,0
2680 NEXT I
2700 RETURN
2720 REM ROUTINE:INIZIALIZZAZIONE COSTANTI
2760 VI=53248:REM CIRCUITO VIDEO
2800 SI=54272:REM CIRCUITO SUONO
2840 MV=1024 :REM MEMORIA VIDEO
2880 MC=55296:REM MEMORIA COLORE
2920 ZL=9:REM COSTANTI DI USO COMUNE
2940 REM INIZIALIZZAZIONE CHIP SUONO
2960 FOR I=0 TO 24
2970 POKE SI+I,0
2980 NEXT I
2990 RETURN
3010 REM ROUTINE:STAMPA IL TITOLO DEL PROGRAMMA MEMORIZZATO IN PG$
3030 GOSUB 2760
3040 POKE VI+32,15
3050 POKE VI+33,15
3060 PRINT "▒▒▒▒";
3070 PRINT TAB(6);" ┌──────────────────────────┐"
3080 FOR I=1 TO 5
3090 PRINT TAB(6);"│                          │"
3100 NEXT I
3110 PRINT TAB(6);" └──────────────────────────┘"
3120 PRINT TAB(6);"▒▒▒▒▒▒▒DIGITA ▒RETURN▒ PER PROSEGUIRE"
3130 PRINT "▒▒▒▒▒"
3140 FOR I=1 TO 5
3150 PRINT TAB(7);
3160 FOR J=1 TO 26
3170 PRINT "▒▒ ";
3180 NEXT J
3190 PRINT
3200 NEXT I
3220 REM ORA SCRIVO IL TITOLO
3240 PRINT "▒▒▒▒▒▒▒▒";TAB((40-LEN(PG$))/2);"▒▒";PG$
3250 GET Z9$
3260 IF Z9$<>CHR$(13) THEN GOTO 3250
3270 PRINT "▒▒";
3280 RETURN
```

Proviamo con l'Assembler

Come sappiamo un calcolatore non è capace di comprendere direttamente le frasi che gli vengono sottoposte, in quanto i suoi circuiti elettronici sono in grado di elaborare solo segnali in codice binario. Quindi se si scrive un programma in Basic o in un qualsiasi altro linguaggio simbolico, un interprete (programma di sistema) dovrà tradurre ciascuna istruzione nei corrispondenti codici binari (linguaggio macchina) che il microprocessore del computer è in grado di comprendere ed eseguire.
Ogni istruzione in linguaggio macchina è formata da un codice operativo (che fornisce al microprocessore l'operazione da eseguire) ed eventualmente da uno operando (in generale un indirizzo di memoria in cui sono contenuti i dati ai quali fa riferimento l'istruzione). Risulta evidente che se si programmasse direttamente in linguaggio macchina si aumenterebbe di molto la velocità di esecuzione del programma, in quanto verrebbe eliminata la fase di traduzione. I programmi dovrebbero però essere scritti in forma binaria e risulterebbero di difficile comprensione in quanto formati da lunghe sequenze di cifre 0 e 1. Per mantenere i vantaggi della programmazione in linguaggio macchina, e nello stesso tempo conservare la massima comprensibilità, si fa uso allora di una rappresentazione delle istruzioni che associa ad ogni codice operativo del linguaggio macchina un codice mnemonico. Si può così programmare usando il linguaggio formato con i codici mnemonici, chiamato Assembler, che si può considerare un linguaggio intermedio tra i linguaggi ad alto livello (quali il Basic, il Pascal, il Cobol, il Fortran, ecc.) e il linguaggio macchina.

Analisi del problema

Il programma che vogliamo scrivere avrà il compito di tradurre in Assembler le istruzioni scritte in linguaggio macchina. Fornendo infatti un indirizzo di memoria saranno visualizzati il codice operativo, l'operando e la corrispondente istruzione in linguaggio assemblativo contenuti nelle locazioni di memoria a partire da quella specificata. Sarà quindi possibile rappresentare le istruzioni Assembler corrispondenti alle istruzioni di un programma scritto in linguaggio macchina, rendendone così più facile la comprensione. Dato che generalmente si fa riferimento agli indirizzi di memoria facendo uso della notazione esadecimale, il programma sarà scritto in modo da accettare indirizzi codificati secondo questa notazione.
Esaminiamo quindi le caratteristiche fondamentali del sistema di numerazione esadecimale, che è un sistema posizionale come quello decimale. Un sistema di numerazione si dice posizionale quando i simboli in esso utilizzati (le cifre da 0 a 9 in quello decimale) assumono un significato diverso in funzione della posizione che hanno all'interno del numero. Nel sistema decimale ad esempio ogni numero viene diviso in unità, decine, centinaia, ecc. In un qualsiasi sistema posizionale per determinare il numero rappresentato dobbiamo prendere le cifre che lo costituiscono da destra verso sinistra, moltiplicarle per le potenze crescenti (potenza zero per la prima cifra a destra) del numero che rappresenta la base del sistema e sommare tra loro i risultati di queste moltiplicazioni. Ad esempio, il valore del numero 3479 in rappresentazione decimale (base 10), è dato dall'espressione:

$$(3\cdot 10^3)+(4\cdot 10^2)+(7\cdot 10^1)+(9\cdot 10^0)=(3\cdot 1000)+(4\cdot 100)+(7\cdot 10)+(9\cdot 1) = \\ =3000+400+70+9=3479$$

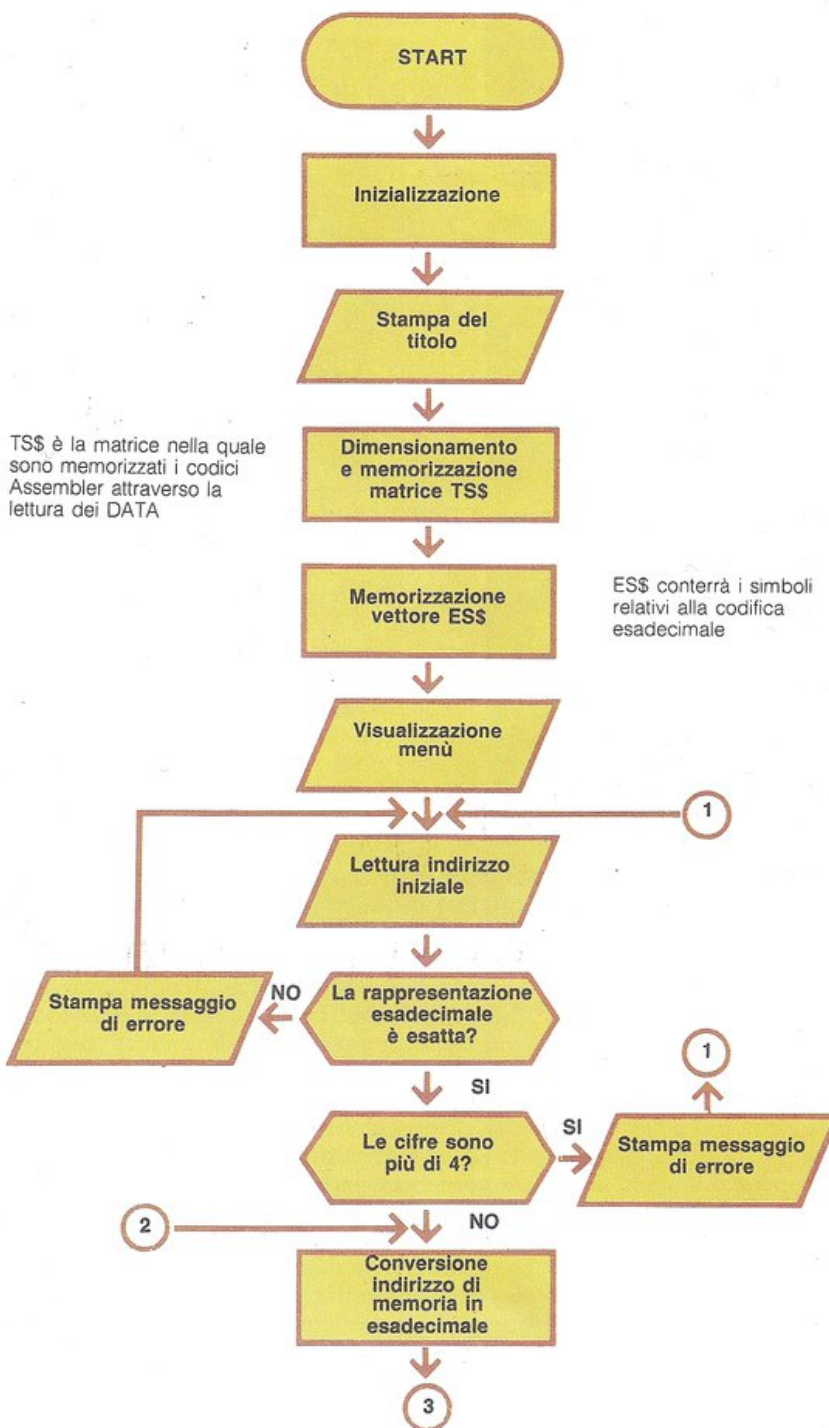
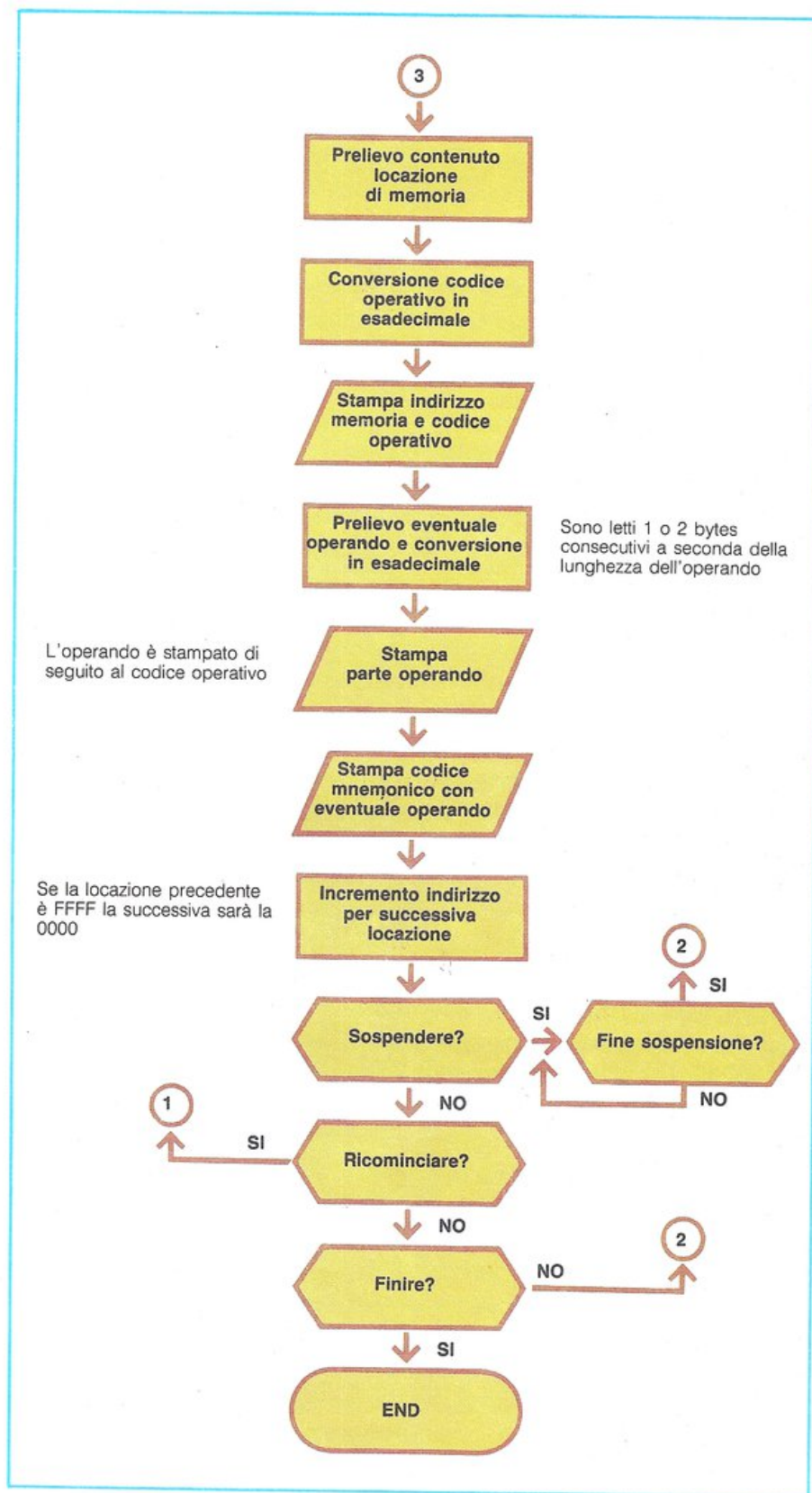
Il sistema esadecimale è concettualmente del tutto simile a quello decimale, solo che viene scelto come base del sistema il numero 16 anziché il 10 e come simboli le lettere A (=10), B(=11), C(=12), D(=13), E(=14),

F(=15) oltre alle cifre da 0 a 9 (16 simboli in tutto). Il valore del numero esadecimale A3F, ad esempio, è dato dall'espressione: $(A \cdot 16^2)+(3 \cdot 16^1)+$ $+(F \cdot 16^0)$ e corrisponderà quindi al numero decimale

$$(10 \cdot 16^2)+(3 \cdot 16^1)+(15 \cdot 16^0)=2560+48+15= 2623$$

Se si vuole ottenere il passaggio inverso, ovvero convertire un numero decimale in esadecimale, si può usare il metodo delle divisioni successive, che consiste nel dividere più volte il numero decimale per sedici, fino ad ottenere un quoziente avente la parte intera nulla, e annotando ogni volta il resto. Per esempio, dovendo convertire il numero decimale 949 si dovranno effettuare le seguenti divisioni: 949/16= 59 con resto 5; 59/16=3 con resto 11 (11=B in esadecimale); 3/16= 0 con resto 3; il numero esadecimale corrispondente a 949 è quindi 3B5.
Per tornare al programma, innanzitutto sarà necessario avere a disposizione tutte le istruzioni Assembler associate alle istruzioni in linguaggio macchina. A tale scopo sarà utilizzata una matrice di 256 righe per tre colonne. La scelta delle dimensioni della matrice non è casuale: infatti, poiché i codici operativi delle istruzioni macchina del nostro microprocessore sono lunghi un solo byte (8 bits), se ne possono avere al massimo 256 (2^8). In realtà i codici operativi effettivamente utilizzati sono solo 57; i 199 valori che restano non sono significativi, e nella matrice vengono rappresentati con tre asterischi.
Il programma leggerà le informazioni da inserire nella matrice dalle apposite istruzioni DATA. Ogni istruzione in linguaggio macchina, come si è detto, può includere un operando; è quindi prevista nei DATA, per ogni istruzione Assembler, l'indicazione della lunghezza in bytes (al massimo 2) dell'operando e della posizione in cui dovrà essere visualizzato (la matrice prevede infatti 3 colonne per memorizzare, oltre al codice mnemonico, anche queste due informazioni). Ad esempio, ADC $ 1,5 indica che l'istruzione in linguaggio macchina corrispondente ha un operando di lunghezza pari ad un byte che deve essere posizionato nel sesto carattere a partire da sinistra (cioè dopo ADC $). Se ad esempio in una coppia di celle di memoria contigue è contenuta l'istruzione in linguaggio macchina 654A (di cui 4A è la parte operando), la corrispondente istruzione Assembler sarà visualizzata come ADC $4A.
Sarà inoltre necessario inizializzare un vettore contenente i simboli del sistema esadecimale, che servirà per convertire il numero introdotto nel corrispondente decimale. Infatti, poiché il programma è scritto in Basic, bisognerà convertire in notazione decimale ogni indirizzo di memoria fornito, prima di trovare (con una PEEK) il codice operativo dell'istruzione corrispondente in linguaggio macchina. Si procederà poi a leggere nella matrice l'istruzione Assembler associata, ad inserirvi, secondo le modalità previste, la parte operando qualora fosse presente, e a visualizzare tutte queste informazioni. La riga di stampa sarà quindi composta in sequenza: indirizzo della locazione di memoria, codice operativo con eventuale operando, codice mnemonico con eventuale operando.
Il programma continuerà fornendo i dati relativi agli indirizzi di memoria successivi a quello introdotto (ricominciando dall'inizio dopo aver visualizzato il contenuto dell'ultima memoria indirizzabile), fino a che l'utente non digiti un comando da tastiera. La locazione di memoria ad indirizzo più elevato nel CBM-64 è la 65535; infatti 64 kbytes corrispondono ad un totale di 64 * 1024 = 65536 bytes e, considerando lo zero, la rappresentazione esadecimale dell'ultima locazione è FFFF (=65535 decimale). FFFF sarà quindi il massimo indirizzo accettato dal programma.
I comandi disponibili saranno i seguenti: R per interrompere l'esecuzione e fornire un nuovo indirizzo di partenza, F per porre fine al programma ed infine S per sospendere momentaneamente l'esecuzione che riprenderà non appena sarà rilasciato il tasto.

Diagrammi di flusso

In testa è presente la solita fase di stampa del titolo e di inizializzazione delle costanti. Si prosegue con il dimensionamento e la memorizzazione della matrice TS$ attraverso la lettura dei DATA. La matrice contiene ora i simboli (codici mnemonici) che il programma andrà ad associare al corrispondente indirizzo di memoria.
Sono poi trasferiti nel vettore ES$ i simboli relativi alla codifica esadecimale. Quindi, dopo aver presentato il menù dei comandi, il programma richiede da quale locazione l'utente intenda iniziare il disassemblamento. Verifica se l'indirizzo immesso è rappresentato in esadecimale e se è composto al massimo da quattro cifre: in caso negativo torna a richiedere l'indirizzo, dopo aver stampato i relativi messaggi d'errore, mentre in caso positivo continua usando il vettore di trasformazione (ES$) per tradurre l'indirizzo in decimale.
A questo punto l'indirizzo della locazione è rappresentato in decimale, e il programma va a leggere il suo contenuto tramite la funzione di sistema PEEK.
Il blocco successivo trasforma il valore letto (che è in forma decimale) in una stringa esadecimale, che è poi stampata insieme all'indirizzo della locazione di memoria.
Si preleva quindi l'eventuale operando, che

può essere composto da 1 o 2 bytes, dalle locazioni di memoria immediatamente successive a quella letta in precedenza. Anche questo è tradotto in notazione esadecimale, ed è poi stampato di seguito al codice operativo. La riga di stampa è completata dal codice mnemonico, unito all'eventuale operando. Per proseguire nel disassemblamento si va poi a calcolare l'indirizzo della locazione successiva, dopo di che il programma esamina se è stato digitato un comando da tastiera. Il primo è il comando di sosta che permette di sospendere l'esecuzione e la stampa fino a quando si mantenga premuto il tasto S. Il secondo è il comando di uscita dal ciclo (tasto R) che fa tornare al punto del programma in cui si richiede l'indirizzo della prima locazione. L'ultimo è il comando che permette di terminare definitivamente l'esecuzione del programma (tasto F).
Se non è stato premuto alcun tasto, oppure se ne è stato premuto uno differente da quelli indicati, si passa a considerare la locazione di memoria che segue, riprendendo il ciclo dalla conversione dell'indirizzo in esadecimale.

Il programma

Le istruzioni contenute nelle linee dalla 120 alla 150 permettono di visualizzare il titolo del programma e di dimensionare la matrice TS$ e il vettore ES$. Dopo aver inviato un messaggio all'utente (linee 211, 212) nel quale si prega di attendere, il programma memorizza nella matrice TS$ (linee 220÷240) le informazioni relative alle istruzioni Assembler leggendole dai DATA delle linee 30000÷30225, operazione che richiede un tempo relativamente lungo. Sono poi inseriti nel vettore ES$ i simboli utilizzati nella numerazione esadecimale (linee 243÷248). Dopo la stampa del menù dei comandi (linee 263÷275), la linea 290 passa il controllo alla routine 10000 che, chiesto all'utente l'indirizzo di memoria da cui vuole iniziare il disassemblaggio, verifica che il numero digitato sia esadecimale (linee 10030÷10110) e, qualora non lo fosse, stampa un messaggio di avvertimento e si riposiziona in attesa di un indirizzo. Quando invece il numero digitato è corretto questa routine, tolti eventuali zeri presenti all'inizio del numero, in quanto non significativi (linea 10113), ne verifica la lunghezza e, nel caso che questa superi le quattro cifre, visualizza il messaggio: «attenzione, gli indirizzi a disposizione vanno da 0000 a FFFF» e attende un nuovo indirizzo.

```
0 REM ****************************
1 REM *PROVIAMO CON L'ASSEMBLER*
2 REM ****************************
4 REM VARIABILI UTILIZZATE
6 REM I,J          :CONTATORI DI CICLO
7 REM SD           :INDIRIZZO DELLA LOCAZIONE DA DISASSEMBLARE
8 REM SD$          :INDIRIZZO ESADECIMALE DELLA LOCAZIONE DI MEM. DA DISASSEMBLARE
9 REM C$           :USATA NELLA ROUTINE DI INPUT
10 REM CN          :VALORE DA CONVERTIRE IN ESADECIMALE
11 REM CN$         :STRINGA ESADECIMALE OTTENUTA DALLA CONVERSIONE DI CN
12 REM RE          :USATA NELLA ROUTINE DI CONVERSIONE DECIMALE-ESADECIMALE
13 REM OP          :CODICE OPERATIVO DELL'ISTRUZIONE DA DISASSEMBLARE
14 REM LU          :LUNGHEZZA DELLA PARTE OPERANDO DELL'ISTR. DA DISASSEMBLARE
15 REM IL$         :PARTE BASSA DELLA PARTE OPERANDO, IN ESADECIMALE
16 REM IH$         :PARTE ALTA DELLA PARTE OPERANDO, IN ESADECIMALE
17 REM OP$         :ASCII DEL CODICE OPERATIVO
18 REM PO          :POSIZIONE,NELLA STRINGA CODICE OPERATIVO,DELLA PARTE OPERANDO
19 REM PI$         :STRINGA CONTENENTE L'INTERA PARTE OPERANDO
20 REM A$          :USATA PER SVUOTARE IL BUFFER DI INGRESSO
21 REM A           :CONTIENE IL COMANDO IMMESSO DA TASTIERA
22 REM IL          :DECIMALE DI IL$
23 REM IH          :DECIMALE DI IH$
24 REM ES$(15)     :USATO NELLA CONVERSIONE DECIMALE-ESADECIMALE ED INVERSA
25 REM TS$(255,2):CONTIENE I CODICI MNEMONICI DELLE ISTRUZIONI
110 REM TITOLO ED INIZIALIZZAZIONE COSTANTI C64
120 PG$="PROVIAMO CON L'ASSEMBLER"
130 GOSUB 62000
140 REM DIMENSIONO ARRAY
150 DIM TS$(255,2),ES$(15)
160 REM LEGGO LA TAVOLA DEI SIMBOLI MEMORIZZATA CON ISTRUZIONI DATA
170 REM LA TAVOLA DEI SIMBOLI E' COMPOSTA DA 256 RIGHE DI 3 COLONNE
180 REM NELLA PRIMA COLONNA C'E' IL CODICE MNEMONICO SENZA PARTE INDIRIZZO
190 REM NELLA SECONDA C'E' LA LUNGHEZZA DELLA PARTE INDIRIZZO; 1 O 2 BYTES
200 REM NELLA TERZA COLONNA C'E' LA POSIZIONE DELLA PARTE INDIRIZZO
210 REM ALL'INTERNO DELLA PARTE CODICE MNEMONICO
211 PRINT "▒▒▒ATTENDERE UN MOMENTO,"
212 PRINT "▒PER FAVORE."
220 FOR I=0 TO 255
230 READ TS$(I,0),TS$(I,1),TS$(I,2)
240 NEXT I
241 REM INIZIALIZZO IL VETTORE USATO NELLA CONVERSIONE DECIMALE-ESADECIMALE
243 FOR I=0 TO 9
244 ES$(I)=RIGHT$(STR$(I),1)
245 NEXT I
246 FOR I=10 TO 15
247 ES$(I)=CHR$(ASC("A")+I-10)
248 NEXT I
250 REM INIZIA IL PROGRAMMA DISASSEMBLATORE
260 REM CARATTERI IN NERO
261 PRINT "▒";
263 REM VISUALIZZO IL MENU
270 PRINT"▒▒▒"
271 PRINT"▒SE VUOI SOSPENDERE TIENI PREMUTO S"
273 PRINT"▒SE VUOI RICOMINCIARE PREMI R"
275 PRINT"▒SE VUOI FINIRE PREMI F"
280 REM CHIEDO LOCAZIONE DA CUI INIZIARE IL DISASSEMBLAGGIO
290 GOSUB 10000
300 REM LA LOCAZIONE DA CUI INIZIARE E' MEMORIZZATA IN SD
330 REM INIZIO IL DISASSEMBLAGGIO
340 PRINT "▒▒";
350 REM CONVERTO SD IN UNA STRINGA ESADECIMALE DI QUATTRO CIFRE
360 CN=SD
370 GOSUB 20000
380 REM STAMPO INDIRIZZO DI MEMORIA
390 PRINT CN$;
400 REM STAMPO LA PARTE CODICE OPERATIVO
410 OP=PEEK(SD)
420 CN=OP
430 GOSUB 20000
440 PRINT TAB(6);RIGHT$(CN$,2);
450 REM LUNGHEZZA DELLA PARTE OPERANDO
460 LU=VAL(TS$(OP,1))
470 REM CARICO LA PARTE OPERANDO IN IL$,IH$
480 GOSUB 21000
490 REM STAMPO L'OPERANDO DI SEGUITO AL CODICE OPERATIVO
500 PRINT IL$;IH$;
510 REM MANIPOLO IL CODICE OPERATIVO
520 OP$=TS$(OP,0):IF OP$=""THENOP$="***"
```

Qualora la lunghezza del numero fornito non superi quella massima prevista il numero è convertito in decimale facendo riferimento al vettore ES$ (linee 10180÷10270). Il programma poi memorizza il valore così ottenuto nella variabile CN, chiama la routine 20000, che lo riconverte in una stringa esadecimale di quattro cifre (linee 20030÷20090), e lo visualizza (linea 390). La conversione da esadecimale in decimale è necessaria per leggere il contenuto della locazione di memoria con la funzione PEEK (linea 410), che trasferisce nella variabile OP il codice operativo (in notazione decimale). Il valore di OP è poi assegnato a CN (linea 420) ed è chiamata di nuovo la routine 20000 (linea 430) che lo converte in esadecimale per poterlo visualizzare (linea 440). Nella linea 460 si determina la lunghezza dell'operando (0, 1 o 2 bytes) e successivamente (linea 480) si chiama la routine 21000, che memorizza il valore del 1° byte in IL$ e quello dell'eventuale 2° byte in IH$, convertendoli poi in esadecimale utilizzando ancora la routine 20000. Sono quindi stampati IL$ e IH$, che non conterranno alcun valore nel caso che non sia presente alcun operando associato al codice operativo (linea 500). Si carica poi nella stringa OP$ l'istruzione Assembler corrispondente al codice operativo già determinato, inserendo tre asterischi nel caso che questa non esista (linea 520); quindi è riportata nella variabile PO la posizione

```
530 REM POSIZIONE DELLA PARTE OPERANDO
540 PO=VAL(TS$(OP,2))
550 REM CONTROLLO SE LA PARTE OPERANDO E' COMPOSTA DA 1 O 2 BYTES
560 PI$=IL$
570 IF LU=2 THEN PI$=IH$+IL$
580 REM AGGIUNGO LA PARTE OPERANDO ALLA STRINGA OP$
590 OP$=LEFT$(OP$,PO)+PI$+RIGHT$(OP$,LEN(OP$)-PO)
600 REM STAMPO CODICE MNEMONICO
610 PRINT TAB(15);OP$
620 REM E' TERMINATO IL DISASSEMBLAGGIO DI UNA LINEA DI PROGRAMMA
630 REM CAMBIO LINEA
640 SD=SD+LU+1
650 REM L'INDIRIZZO NON PUO' SUPERARE IL VALORE DI (2↑16-1)
660 SD=SD-INT(SD/2↑16)*2↑16
670 REM ORA ABBIAMO A DISPOSIZIONE TRE COMANDI:
680 REM S:SOSPENDE TEMPORANEAMENTE LA STAMPA
690 REM R:SI VUOLE DISASSEMBLARE DA UN NUOVO INDIRIZZO
700 REM F:FINE PROGRAMMA
701 REM PULISCO IL BUFFER DI INGRESSO
702 FOR I=1 TO 10:GET A$:NEXT I
710 A=PEEK(197)
720 IF A=13 THEN GOTO 710
730 IF A=17 THEN GOTO 290
740 IF A=21 THEN PRINT "▒▒▒":GOSUB21300 :END
750 GOTO 360
760 REM INIZIANO LE ROUTINES
9990 REM ROUTINE:CHIEDE LA LOCAZIONE DA CUI INIZIARE A DISASSEMBLARE
10000 INPUT "▒▒INDIRIZZO, IN ESADECIMALE ";SD$
10010 REM CONTROLLO CHE IL  NUMERO SIA EFFETTIVAMENTE ESADECIMALE
10020 IF SD$="" THEN SD=0:RETURN
10030 I=1
10040 C$=MID$(SD$,I,1)
10050 IF (C$>="0" AND C$<="9") OR (C$>="A" AND C$<="F") THEN GOTO 10100
10060 REM LA CIFRA NON E' ACCETTABILE
10070 PRINT "▒ATTENZIONE, UNA CIFRA ESADECIMALE"
10080 PRINT "NON PUO' CONTENERE IL CARATTERE ";C$
10090 GOTO 10000
10100 I=I+1
10110 IF I<=LEN(SD$) THEN GOTO 10040
10111 REM TOLGO GLI EVENTUALI ZERI NON SIGNIFICATIVI IN TESTA ALLA CIFRA
10112 IF LEN(SD$)=1 THEN GOTO 10120
10113 IF LEFT$(SD$,1)="0" THEN SD$=RIGHT$(SD$,LEN(SD$)-1):GOTO 10113
10114 IF SD$="" THEN GOTO 10020
10120 REM IL NUMERO E' CORRETTO. CONTROLLO CHE NON SUPERI LE 4 CIFRE
10130 IF LEN(SD$)<=4 THEN GOTO 10180
10140 PRINT "▒ATTENZIONE, GLI INDIRIZZI A DISPOSIZIONE"
10150 PRINT "▒VANNO DA 0000 A FFFF"
10160 GOTO 10000
10170 REM ORA POSSO CONVERTIRE LA CIFRA SD$ IN DECIMALE
10180 SD=0
10190 FOR I=LEN(SD$) TO 1 STEP -1
10200 REM CERCO IL CARATTERE IN ES$(15)
10210 J=0
10220 IF ES$(J)=MID$(SD$,I,1) THEN GOTO 10260
10230 J=J+1
10240 GOTO 10220
10250 REM CARATTERE TROVATO
10260 SD=SD+J*16↑(LEN(SD$)-I)
10270 NEXT I
10280 RETURN
19990 REM ROUTINE:CONVERTE LA CIFRA IN CN IN UNA STRINGA ESADECIMALE DI 4 CIFRE
20000 I=1
20010 CN$=""
20020 IF CN=0 THEN GOTO 20080
20030 RE=CN-INT(CN/16)*16
20040 CN$=ES$(RE)+CN$
20050 CN=INT(CN/16)
20060 I=I+1
20070 GOTO 20020
20080 IF I<5 THEN CN$="0"+CN$:I=I+1:GOTO 20080
20090 REM CONVERSIONE TERMINATA
20100 RETURN
20990 REM ROUTINE:CARICA LA PARTE OPERANDO NELLE DUE STRINGHE IH$,IL$
21000 IL$="":IH$=""
21010 IF LU=0 THEN RETURN
21020 IL=PEEK(SD+1)
21030 REM CONVERTO IL PRIMO,EVENTUALMENTE UNICO BYTE DI INDIRIZZO,IN ESADECIMALE
21040 CN=IL
```

che dovrà assumere l'operando di tale istruzione (linea 540). Si dovrà poi controllare da quanti bytes è composto l'operando (linea 560, 570), aggiungerlo alla stringa OP$ nella giusta posizione (linea 590) e visualizzare infine il contenuto di OP$, cioè l'istruzione Assembler con il relativo operando (linea 610).
Terminato così il disassemblaggio di una istruzione, si passa all'indirizzo della successiva (linea 640), si controlla che non sia maggiore di $2^{16}-1$ (che corrisponde all'esadecimale FFFF) e, nel caso lo fosse, lo si pone pari a 0 (linea 660). Il programma continua la sua esecuzione fino a quando non sia immesso da tastiera uno dei seguenti comandi: S per interrompere momentaneamente l'esecuzione (linee 710, 720), R per introdurre un nuovo indirizzo di partenza (linea 730) e F per terminare l'esecuzione del programma (linea 740).

```
21050 GOSUB 20000
21055 IL$=RIGHT$(CN$,2)
21060 REM CONTROLLO SE C'E' UN SECONDO BYTE DI INDIRIZZO
21070 IF LU=1 THEN RETURN
21080 IH=PEEK(SD+2)
21090 CN=IH
21100 GOSUB 20000
21105 IH$=RIGHT$(CN$,2)
21110 RETURN
21200 REM ROUTINE RITORNO AI COLORI NORMALI
21300 POKE53280,14:POKE53281,6
21310 PRINT"□";
21320 PRINT"□"
21400 RETURN
29990 REM TAVOLA DEI SIMBOLI
30000 DATA BRK,0,0
30001 DATA "ORA ($,X)",1,6
30002 DATA ***,0,0
30003 DATA ***,0,0
30004 DATA ***,0,0
30005 DATA ORA $,1,5
30006 DATA ASL $,1,5
30007 DATA ***,0,0
30008 DATA PHP,0,0
30009 DATA ORA #$,1,6
30010 DATA ASL,0,0
30011 DATA ***,0,0
30012 DATA ***,0,0
30013 DATA ORA $,2,5
30014 DATA ASL $,2,5
30015 DATA ***,0,0
30016 DATA BPL $,1,5
30017 DATA "ORA ($),Y",1,6
30018 DATA ***,0,0
30019 DATA ***,0,0
30020 DATA ***,0,0
30021 DATA "ORA $,X",1,5
30022 DATA "ASL $,X",1,5
30023 DATA ***,0,0
30024 DATA CLC,0,0
30025 DATA "ORA $,Y",2,5
30026 DATA ***,0,0
30027 DATA ***,0,0
30028 DATA ***,0,0
30029 DATA "ORA $,X",2,5
30030 DATA "ASL $,X",2,5
30031 DATA ***,0,0
30032 DATA JSR $,2,5
30033 DATA "AND ($,X)",1,6
30034 DATA ***,0,0
30035 DATA ***,0,0
30036 DATA BIT $,1,5
30037 DATA AND $,1,5
30038 DATA ROL $,1,5
30039 DATA ***,0,0
30040 DATA PLP,0,0
30041 DATA AND #$,1,6
30042 DATA ROL,0,0
30043 DATA ***,0,0
30044 DATA BIT $,2,5
30045 DATA AND $,2,5
30046 DATA ROL $,2,5
30047 DATA ***,0,0
30048 DATA BMI $,1,5
30049 DATA "AND ($),Y",1,6
30050 DATA ***,0,0
30051 DATA ***,0,0
30052 DATA ***,0,0
30053 DATA "AND $,X",1,5
30054 DATA "ROL $,X",1,5
30055 DATA ***,0,0
30056 DATA SEC,0,0
30057 DATA "AND $,Y",2,5
30058 DATA ***,0,0
30059 DATA ***,0,0
30060 DATA ***,0,0
30061 DATA "AND $,X",2,5
30062 DATA "ROL $,X",2,5
```

```
30063 DATA ***,0,0
30064 DATA RTI,0,0
30065 DATA "EOR ($),Y",1,6
30066 DATA ***,0,0
30067 DATA ***,0,0
30068 DATA ***,0,0
30069 DATA EOR $,1,5
30070 DATA LSR $,1,5
30071 DATA ***,0,0
30072 DATA PHA,0,0
30073 DATA EOR #$,1,6
30074 DATA LSR,0,0
30075 DATA ***,0,0
30076 DATA JMP $,2,5
30077 DATA EOR $,2,5
30078 DATA LSR $,2,5
30079 DATA ***,0,0
30080 DATA BVC $,1,5
30081 DATA "EOR ($),Y",1,6
30082 DATA ***,0,0
30083 DATA ***,0,0
30084 DATA ***,0,0
30085 DATA "EOR $,X",1,5
30086 DATA "LSR $,X",1,5
30087 DATA ***,0,0
30088 DATA CLI,0,0
30089 DATA "EOR $,Y",2,5
30090 DATA ***,0,0
30091 DATA ***,0,0
30092 DATA ***,0,0
30093 DATA "EOR $,X",2,5
30094 DATA "LSR $,X",2,5
30095 DATA ***,0,0
30096 DATA RTS,0,0
30097 DATA "ADC ($,X)",1,6
30098 DATA ***,0,0
30099 DATA ***,0,0
30100 DATA ***,0,0
30101 DATA ADC $,1,5
30102 DATA ROR $,1,5
30103 DATA ***,0,0
30104 DATA PLA,0,0
30105 DATA ADC #$,1,6
30106 DATA ROR,0,0
30107 DATA ***,0,0
30108 DATA JMP ($),2,6
30109 DATA ADC $,2,5
30110 DATA ROR $,2,5
30111 DATA ***,0,0
30112 DATA BVS $,1,5
30113 DATA "ADC ($),Y",1,6
30114 DATA ***,0,0
30115 DATA ***,0,0
30116 DATA ***,0,0
30117 DATA "ADC $,X",1,5
30118 DATA "ROR $,X",1,5
30119 DATA ***,0,0
30120 DATA SEI,0,0
30121 DATA "ADC $,Y",2,5
30122 DATA ***,0,0
30123 DATA ***,0,0
30124 DATA ***,0,0
30125 DATA "ADC $,X",2,5
30126 DATA "ROR $,X",2,5
30127 DATA ***,0,0
30128 DATA ***,0,0
30129 DATA "STA ($,X)",1,6
30130 DATA ***,0,0
30131 DATA ***,0,0
30132 DATA STY $,1,5
30133 DATA STA $,1,5
30134 DATA STX $,1,5
30135 DATA ***,0,0
30136 DATA DEY,0,0
30137 DATA ***,0,0
30138 DATA TXA,0,0
30139 DATA ***,0,0
30140 DATA STY $,2,5
```

```
30141 DATA STA $,2,5
30142 DATA STX $,2,5
30143 DATA ***,0,0
30144 DATA BCC $,1,5
30145 DATA "STA ($),Y",1,6
30146 DATA ***,0,0
30147 DATA ***,0,0
30148 DATA "STY $,X",1,5
30149 DATA "STA $,X",1,5
30150 DATA "STX $,Y",1,5
30151 DATA ***,0,0
30152 DATA TYA,0,0
30153 DATA "STA $,Y",2,5
30154 DATA TXS,0,0
30155 DATA ***,0,0
30156 DATA ***,0,0
30157 DATA "STA $,X",2,5
30158 DATA ***,0,0
30159 DATA ***,0,0
30160 DATA LDY #$,1,6
30161 DATA "LDA ($,X)",1,6
30162 DATA LDX #$,1,6
30163 DATA ***,0,0
30164 DATA LDY $,1,5
30165 DATA LDA $,1,5
30166 DATA LDX $,1,5
30167 DATA ***,0,0
30168 DATA TAY,0,0
30169 DATA LDA #$,1,6
30170 DATA TAX,0,0
30171 DATA ***,0,0
30172 DATA LDY $,2,5
30173 DATA LDA $,2,5
30174 DATA LDX $,2,5
30175 DATA ***,0,0
30176 DATA BCS $,1,5
30177 DATA "LDA ($),Y",1,6
30178 DATA ***,0,0
30179 DATA ***,0,0
30180 DATA "LDY $,X",1,5
30181 DATA "LDA $,X",1,5
30182 DATA "LDX $,Y",1,5
30183 DATA ***,0,0
30184 DATA CLV,0,0
30185 DATA "LDA $,Y",2,5
30186 DATA TSX,0,0
30187 DATA ***,0,0
30188 DATA "LDY $,X",2,5
30189 DATA "LDA $,X",2,5
30190 DATA "LDX $,Y",2,5
30191 DATA ***,0,0
30192 DATA CPY #$,1,6
30193 DATA "CMP ($,X)",1,6
30194 DATA ***,0,0
30195 DATA ***,0,0
30196 DATA CPY $,1,5
30197 DATA CMP $,1,5
30198 DATA DEC $,1,5
30199 DATA ***,0,0
30200 DATA INY,0,0
30201 DATA CMP #$,1,6
30202 DATA DEX,0,0
30203 DATA ***,0,0
30204 DATA CPY $,2,5
30205 DATA CMP $,2,5
30206 DATA DEC $,2,5
30207 DATA ***,0,0
30208 DATA BNE $,1,5
30209 DATA "CMP ($),Y",1,6
30210 DATA ***,0,0
30211 DATA ***,0,0
30212 DATA ***,0,0
30213 DATA "CMP $,X",1,5
30214 DATA "DEC $,X",1,5
30215 DATA ***,0,0
30216 DATA CLD,0,0
30217 DATA "CMP $,Y",2,5
30218 DATA ***,0,0
```

```
30219 DATA ***,0,0
30220 DATA ***,0,0
30221 DATA "CMP $,X",2,5
30222 DATA "DEC $,X",2,5
30223 DATA ***,0,0
30224 DATA CPX #$,1,6
30225 DATA "SBC ($,X)",1,6
30226 DATA ***,0,0
30227 DATA ***,0,0
30228 DATA CPX $,1,5
30229 DATA SBC $,1,5
30230 DATA INC $,1,5
30231 DATA ***,0,0
30232 DATA INX,0,0
30233 DATA SBC #$,1,6
30234 DATA NOP,0,0
30235 DATA ***,0,0
30236 DATA CPX $,2,5
30237 DATA SBC $,2,5
30238 DATA INC $,2,5
30239 DATA ***,0,0
30240 DATA BEQ $,1,5
30241 DATA "SBC ($),Y",1,6
30242 DATA ***,0,0
30243 DATA ***,0,0
30244 DATA ***,0,0
30245 DATA "SBC $,X",1,5
30246 DATA "INC $,X",1,5
30247 DATA ***,0,0
30248 DATA SED,0,0
30249 DATA "SBC $,Y",2,5
30250 DATA ***,0,0
30251 DATA ***,0,0
30252 DATA ***,0,0
30253 DATA "SBC $,X",2,5
30254 DATA "INC $,X",2,5
30255 DATA ***,0,0
60960 REM ROUTINE INIZIALIZZAZIONE COSTANTI
60980 REM CIRCUITO VIDEO
61000 VI=53248
61020 REM CIRCUITO SUONO
61040 SI=54272
61060 REM MEMORIA VIDEO
61080 MV=1024
61100 REM MEMORIA COLORE
61120 MC=55296
61140 REM COSTANTI DI USO COMUNE
61160 ZL=9
61180 REM INIZIALIZZAZIONE CHIP SUONO
61200 FOR I=0 TO 24
61210 POKE SI+I,0
61220 NEXT I
61230 RETURN
61980 REM ROUTINE:STAMPA IL TITOLO DEL PROGRAMMA MEMORIZZATO IN PG$
62000 GOSUB 61000
62010 POKE VI+32,15
62020 POKE VI+33,15
62030 PRINT "▒▒▒▒▒▒";
62040 PRINT TAB(6);"┌──────────────────────────┐"
62050 FOR I=1 TO 5
62060 PRINT TAB(6);"│                          │"
62070 NEXT I
62080 PRINT TAB(6);"└──────────────────────────┘"
62090 PRINT TAB(6);"▒▒▒▒▒▒▒DIGITA ▒RETURN■ PER PROSEGUIRE"
62100 PRINT "▒▒▒▒▒▒"
62110 FOR I=1 TO 5
62120 PRINT TAB(7);
62130 FOR J=1 TO 26
62140 PRINT "▒▒ ";
62150 NEXT J
62160 PRINT
62170 NEXT I
62190 REM ORA SCRIVO IL TITOLO
62210 PRINT "▒▒▒▒▒▒▒▒▒▒",TAB((40-LEN(PG$))/2);"▒▒";PG$
62220 GET Z9$
62230 IF Z9$<>CHR$(13) THEN GOTO 62220
62240 PRINT "▒▒",
62250 RETURN
```

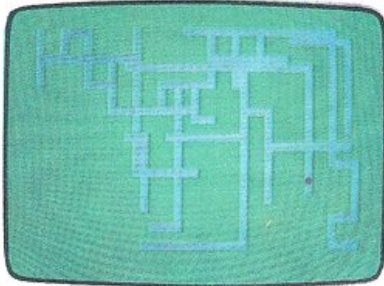
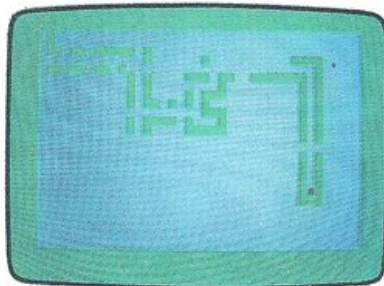
Labirinto

La costruzione di un labirinto è un'operazione comune nella progettazione di molti videogiochi. Il sistema più semplice consiste nel definire il disegno di un labirinto una volta per tutte, assegnando ad ogni punto del video un opportuno carattere. In questo modo però il giocatore si troverà di fronte sempre lo stesso labirinto, e in breve tempo imparerà a conoscerlo perfettamente.
Per evitare questo inconveniente è possibile progettare il programma in modo tale che ad ogni esecuzione sia costruito un labirinto diverso dai precedenti.
Un altro difetto, riguardante soprattutto i piccoli computer, è rappresentato dallo spazio limitato fornito dal video per la visualizzazione del labirinto. Il programma che sarà presentato permetterà di superare anche questa limitazione e di rendere quindi il gioco il più interessante possibile.
Lo scopo didattico di questo programma è quello di mostrare come si possano generare schemi che rispettino certe condizioni (nel caso dei labirinti deve comunque esistere un collegamento tra l'ingresso e l'uscita), e come si possa utilizzare la memoria per immagazzinare un disegno che non è riportato interamente sul video, ma visualizzato parzialmente mano a mano che il giocatore avanza nel labirinto.
Ciò rende più difficile e interessante il gioco, in quanto il giocatore, non potendo vedere l'intero labirinto, non si può rendere conto se si trova su un percorso che gli permetta di raggiungere l'uscita.

Analisi del problema

Il programma che si vuole realizzare dovrà simulare l'attraversamento di un labirinto. Per rendere più interessante il gioco sarà inoltre prevista la possibilità

- di costruire labirinti sempre diversi
- di rendere l'attraversamento del labirinto più difficile tenendone nascosto lo schema al giocatore e visualizzando solo la strada già percorsa
- di imporre un tempo massimo per raggiungere l'uscita, posta in basso a destra, partendo dall'angolo in alto a sinistra del video.

Il giocatore dovrà avere a disposizione alcuni comandi che gli permettano di spostare il cursore (il quale individua la posizione raggiunta) nella direzione che desidera; qualora il cursore vada ad urtare contro un ostacolo, il giocatore dovrà essere avvertito da un segnale sonoro. Infine, una volta raggiunta l'uscita del labirinto, il programma lo segnalerà facendo lampeggiare il video, se invece il tempo a disposizione finirà prima, dovrà visualizzare l'intero labirinto e la posizione raggiunta dal giocatore. In entrambi i casi sarà proposta all'utente la possibilità di effettuare un altro tentativo.
Lo schema del labirinto, tenuto nascosto al giocatore, sarà rappresentato in memoria; ogni volta che il cursore sarà spostato sul video, l'operazione sarà ripetuta sulla rappresentazione dello schema nella memoria. Dalla rappresentazione nascosta sarà possibile determinare se il cursore ha urtato un muro del labirinto e riprodurre questo fatto con il segnale sonoro.
Dovremo quindi costruire una matrice (LA) di dimensioni 25×40 i cui elementi corrisponderanno ai punti del video. In base al codice memorizzato in un elemento della matrice sarà possibile capire se nel punto corrispondente del labirinto c'è un muro o un corridoio percorribile dal cursore. I punti percorribili saranno rappresentati dal codice 32 (blank)

e quelli occupati dal muro dal 160. Questa tecnica, utilizzata per la memorizzazione di disegni nei calcolatori dedicati alla grafica, permette di associare ad ogni punto del video un elemento della matrice; ad esempio LA (3,2) corrisponde al punto del video posto all'incrocio tra la terza riga e la seconda colonna.

Il programma dovrà ora costruire il labirinto. Tutti gli elementi della matrice LA saranno inizialmente posti a 160 (muro). Sarà poi tracciato il percorso di uscita dal labirinto a partire dal primo punto in alto a sinistra che sarà posto a 32. Ogni tre passi, generando casualmente (con l'uso della funzione RND) un numero compreso tra 0 e 9, si sceglierà una delle quattro possibili direzioni (alto, basso, destra, sinistra) dando una leggera preferenza alle direzioni verso destra e verso il basso. Infatti, in base al numero generato si prenderanno le seguenti decisioni:

- numeri 0÷2: sarà posto a 32 l'elemento della matrice a destra dell'elemento attuale (30% dei casi)
- numeri 3÷5: sarà posto a 32 l'elemento sotto a quello attuale (30% dei casi)
- numeri 6 e 7: sarà posto a 32 l'elemento sopra a quello attuale (20% dei casi)
- numeri 8 e 9: sarà posto a 32 l'elemento a sinistra dell'elemento attuale (20% dei casi).

In ogni caso, prima di porre a 32 uno degli elementi della matrice così individuati, sarà necessario accertarsi che l'elemento esista, cioè che non si trovi al di fuori del video (se ad esempio ci si trova già sul margine sinistro non sarà possibile spostarsi a sinistra). Per verificare se effettivamente è stato costruito un percorso che permette di raggiungere l'uscita, si controllerà il punto nell'angolo in basso a destra: quando l'elemento di LA corrispondente a questo punto sarà stato posto a 32 il percorso sarà stato completato. Per rendere poi più complicato il labirinto saranno inseriti, sempre casualmente, almeno altri 10 percorsi alternativi a forma di L, sparsi nel labirinto, che andranno ad intrecciarsi con il percorso precedentemente determinato. Avremo così la certezza di costruire ogni volta un labirinto diverso, perché il programma determina casualmente sia il tracciato del percorso principale, sia il numero e la posizione dei percorsi alternativi.

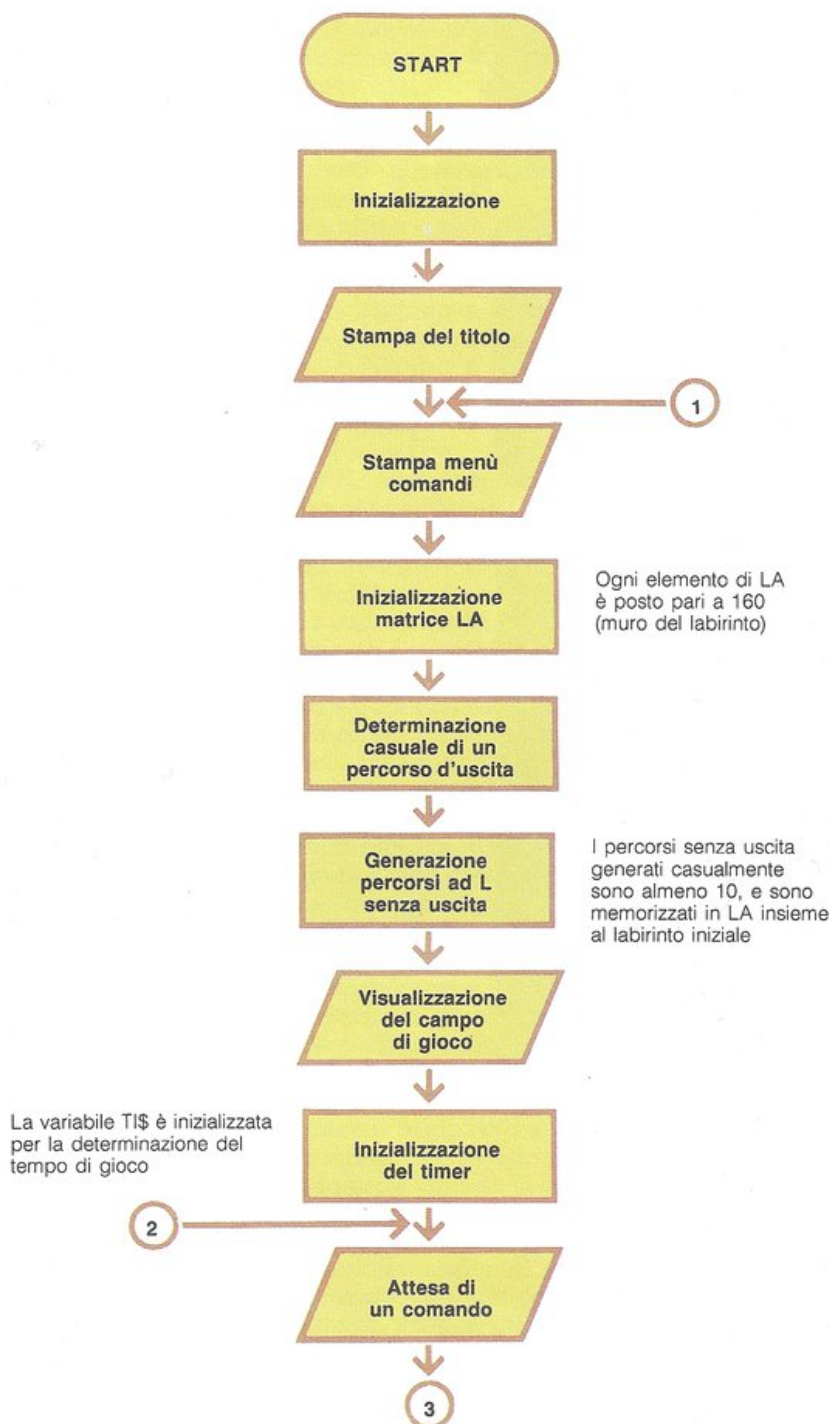
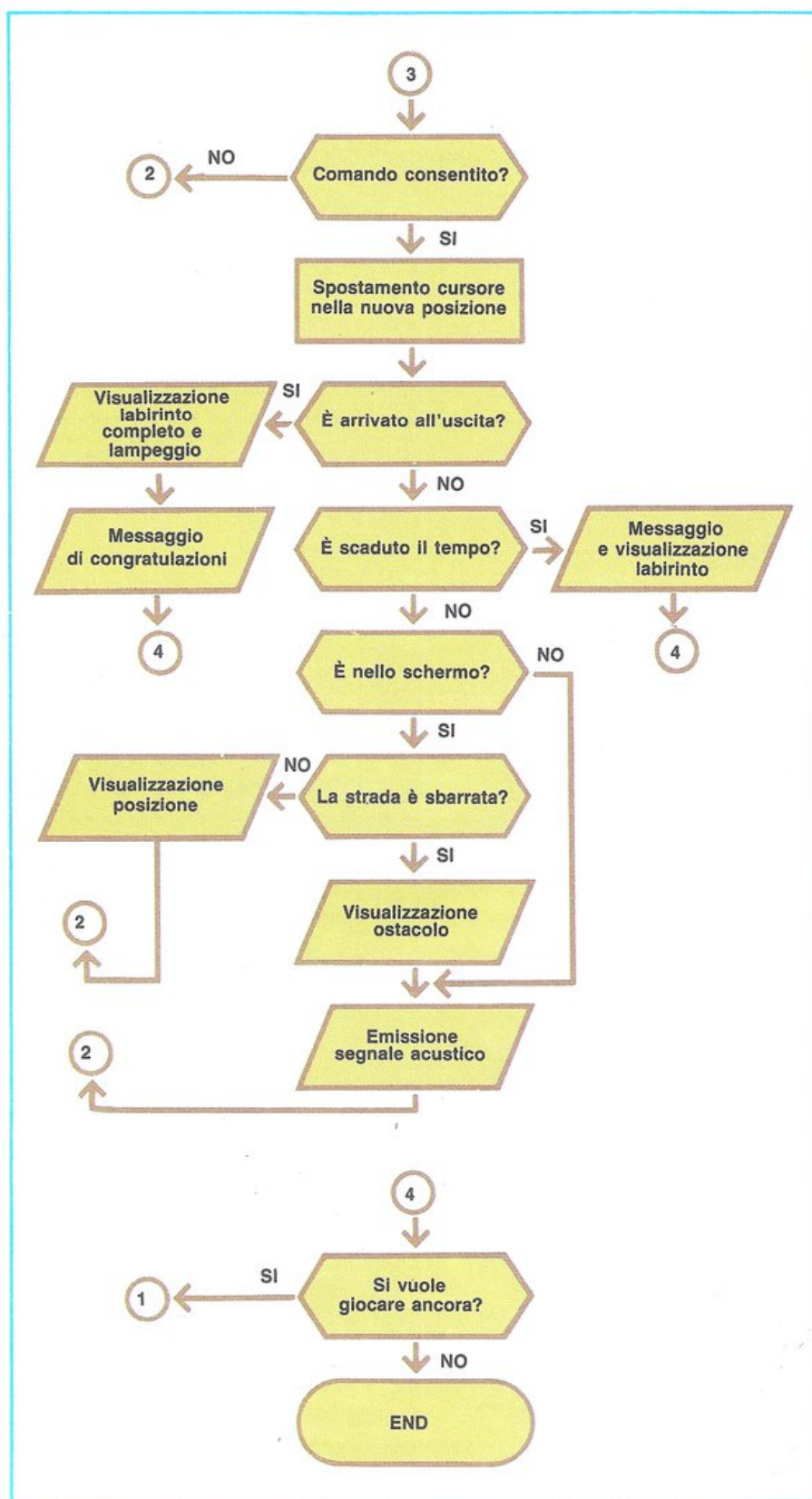
Per tenere nascosto al giocatore lo schema del labirinto il programma provvederà a visualizzare solo gli otto punti che si trovano intorno al cursore mano a mano che questo avanza.

Il tempo massimo per raggiungere l'uscita del labirinto sarà determinato utilizzando la variabile di sistema TI$ legata all'orologio interno. Se tale tempo massimo sarà superato, il programma visualizzerà l'intero labirinto tramite istruzioni cicliche FOR che permetteranno di riportare sul video i valori di ogni elemento della matrice LA. Il lampeggiamento del video, nel caso che il giocatore riesca ad arrivare all'uscita del labirinto, sarà ottenuto sempre con istruzioni cicliche FOR che permetteranno di visualizzare, in rapida successione, la cornice e lo sfondo del video in diversi colori scelti casualmente. Infine i comandi digitati dal giocatore per spostare il cursore saranno riconosciuti, come visto in altri programmi precedenti, in base al codice ASCII del carattere raffigurato sul tasto premuto.

Tali comandi saranno:

- P per spostarsi verso l'alto
- . per spostarsi in basso
- ; per spostarsi a destra
- L per spostarsi a sinistra.

Diagrammi di flusso

Dopo l'inizializzazione delle costanti e la stampa del titolo, il programma presenta il menù dei comandi che il giocatore può utilizzare. Nella fase successiva si memorizza in ogni elemento della matrice LA il numero 160 (muro); si passa poi alla determinazione casuale (e alla memorizzazione) di un percorso (labirinto) che unisce l'ingresso del campo di gioco con l'uscita. Quindi si inizia a generare una serie di percorsi a forma di L, che saranno associati a caso a quello originario; essi rappresentano strade senza uscita, e il loro numero non deve essere inferiore a 10. Si passa poi alla realizzazione del gioco vero e proprio con la stampa del campo di gioco e l'inizializzazione del timer. A questo punto il programma attende che il giocatore digiti un comando per spostare il cursore.

Spostato il cursore si esegue un controllo tendente a verificare se è stata guadagnata l'uscita (questo controllo è naturalmente superfluo all'inizio del gioco, ma servirà per gli spostamenti successivi). Qualora sia stata raggiunta l'uscita è stampato un messaggio ed è visualizzato il labirinto percorso, quindi si domanda al giocatore se intende intraprendere una nuova partita. In caso affermativo, il programma torna a stampare il menù comandi, in caso negativo cessa l'esecuzione.
Ora è necessario determinare il tempo utilizzato dall'inizio del gioco: se è trascorso più di un minuto (tempo massimo concesso) il programma stampa un messaggio col quale avverte che il tempo a disposizione è scaduto, visualizza tutto il labirinto e domanda se si vuole giocare ancora. Nel caso in cui il tempo non sia scaduto, si effettuano due ulteriori controlli.
Con il primo si verifica che il punto determinato sia all'interno dello schermo: se non è così il programma invia un segnale acustico e torna ad attendere un nuovo spostamento. Nel secondo controllo si verifica che la posizione del cursore corrisponda ad un punto percorribile del labirinto; in caso contrario è visualizzato l'ostacolo e si torna ad attendere un nuovo spostamento.
Nel caso in cui i controlli siano superati con esito positivo, si tracciano le pareti del labirinto relative al punto in esame e si torna ad attendere la mossa successiva.

Il programma

Il programma comincia con l'inizializzazione delle costanti e con la presentazione del titolo, utilizzando a tale scopo le routines 61000 e 62000, rispettivamente. Seguono il dimensionamento della matrice LA (linea 140) e la presentazione su video di un messaggio con l'indicazione dell'uscita del labirinto, la lista dei tasti che il giocatore deve utilizzare per effettuare gli spostamenti del cursore e l'avviso che la costruzione del labirinto è in atto (linee 151 ÷ 162). Nelle linee 190 ÷ 230 vi è il doppio ciclo utilizzato per porre il codice 160 in tutti gli elementi della matrice LA. Si entra poi nella fase di costruzione del percorso, ed è quindi inizializzato un generatore casuale (linea 322) utilizzato per determinare la direzione che di volta in volta prenderà il tracciato. Successivamente si stabilisce in 3 il numero minimo dei passi (elementi di LA) del percorso per ogni direzione (linea 324) e si inizializzano gli indici di percorso R e C (linea 330) che conterranno rispettivamente i numeri di riga e di colonna relativi alla posizione del cursore. La linea 340 pone inizialmente a 32 (blank) l'elemento LA (0,0), punto di partenza del tracciato; nel seguito la stessa linea servirà per porre a 32 gli altri punti del percorso che via via saranno determinati. Si incontrano quindi l'istruzione di controllo che

```
0 REM ***********
1 REM *LABIRINTO*
2 REM ***********
4 REM VARIABILI UTILIZZATE
6 REM PG$          :MEMORIZZA IL TITOLO DEL PROGRAMMA
7 REM I,J          :CONTATORI DI CICLO
8 REM DI           :DIREZIONE DEL PERCORSO INGRESSO-USCITA DURANTE LA COSTRUZIONE
9 REM PA           :NUMERO DI PASSI FATTI NELLA ATTUALE DIREZIONE
10 REM R,C         :COORDINATE DEL PUNTO IN ESAME
11 REM R1,C1,R2,C2:COORDINATE DEI VERTICI DELLA L
12 REM A$          :INPUT DA TASTIERA
13 REM RP,RS       :INDICI DELLE RIGHE INTORNO AL CURSORE
14 REM CP,CS       :INDICI DELLE COLONNE INTORNO AL CURSORE
15 REM PO          :INDIRIZZO DEL PUNTO DA ACCENDERE DURANTE LA VISUALIZZAZIONE
16 REM DELL'INTERO LABIRINTO
17 REM A           :INPUT DA TASTIERA
18 REM NR,NC       :COORDINATE DELLA NUOVA POSIZIONE DEL GIOCATORE
19 REM TM          :TEMPO TRASCORSO IN 60-ESIMI DI SECONDO
100 REM TITOLO ED INIZIALIZZAZIONE COSTANTI C64
110 PG$="L A B I R I N T O"
120 GOSUB 62000
130 REM DIMENSIONO ARRAY
140 DIM LA(24,39)
150 REM MESSAGGI PER IL GIOCATORE
151 PRINT "■■";
152 PRINT TAB(10);"L A B I R I N T O"
153 PRINT "■■■L'USCITA DEL LABIRINTO E' NELL'ANGOLO"
154 PRINT "■IN BASSO A DESTRA"
155 PRINT "■■■           PER SPOSTARTI DIGITA:"
156 PRINT "■■P   ALTO"
157 PRINT "■.   BASSO"
158 PRINT "■;   DESTRA"
159 PRINT "■L   SINISTRA"
160 REM MI POSIZIONO SULLA 23-MA RIGA
161 POKE 214,22:PRINT
162 PRINT "■■ATTENDERE, LABIRINTO IN COSTRUZIONE■■"
180 REM IL LABIRINTO E' PRIMA COMPLETAMENTE RIEMPITO DI OSTACOLI.
190 FOR I=0 TO 24
200 FOR J=0 TO 39
210 LA(I,J)=32+128
220 NEXT J
230 NEXT I
240 REM ORA COSTRUISCO IL PERCORSO DALL'ANGOLO IN ALTO A SINISTRA
250 REM ALL'ANGOLO IN BASSO A DESTRA
280 REM IL PERCORSO VIENE COSTRUITO CASUALMENTE RISPETTANDO I SEGUENTI VINCOLI:
290 REM CI SONO QUATTRO DIREZIONI POSSIBILI:ALTO,BASSO,DESTRA
300 REM CON DESTRA,BASSO LEGGERMENTE PIU' PROBABILI DI ALTO E SINISTRA
310 REM 0..2:DESTRA;3..5:BASSO;6..7:ALTO;8..9:SINISTRA
311 REM IN UNA DIREZIONE DEVE ANDARCI ALMENO PER TRE PASSI
320 REM TERMINO QUANDO R=24:C=39
321 REM INIZIALIZZO GENERATORE CASUALE
322 DI=RND(-TI)
323 REM NUMERO DI PASSI =3
324 PA=3
330 R=0:C=0
340 LA(R,C)=32
350 IF R=24 AND C=39 THEN GOTO 460
351 IF PA<3 THEN GOTO 370
352 PA=0
360 DI=INT(RND(DI)*10)
370 IF DI<3 THEN C=C+1:GOTO 410
380 IF DI<6 THEN R=R+1:GOTO 410
381 IF DI<8 THEN R=R-1:GOTO 410
390 C=C-1
400 REM CONTROLLO CHE LA POSIZIONE NON SIA FUORI LABIRINTO
410 IF C=40 THEN C=39
411 IF C=-1 THEN C=0
420 IF R=25 THEN R=24
430 IF R=-1 THEN R=0
431 REM INCREMENTO NUMERO DI PASSI FATTI
432 PA=PA+1
440 GOTO 340
```

permette di sapere se si è finito di costruire il percorso principale (linea 350) e quella che verifica il numero dei passi (linea 351). La direzione del percorso è determinata nelle linee 370÷390 in base al valore assunto dal generatore casuale della linea 360; si verifica poi se tale direzione non comporti l'uscita del tracciato dall'area di costruzione del labirinto (linee 410÷430). Alla linea 432 è incrementato il numero di passi fatti prima di tornare a determinare un nuovo tratto di percorso (linea 440). Finito di costruire il percorso principale, sono selezionati dieci percorsi alternativi a forma di elle. Per fare questo si calcolano casualmente le coordinate dei vertici di una elle (linee 490, 500 e 520, 530) e si costruisce il relativo percorso (linee 550÷620). Si ripetono tali istruzioni fino alla costruzione dei dieci percorsi alternativi (linea 660). Si utilizza poi la funzione RND per generare un numero casuale in base al quale sarà stabilito (con una probabilità del 70%) se costruire ancora una L (linea 680). Quando il numero generato dalla funzione RND è maggiore di 0,7 la costruzione del labirinto ha termine, ed è visualizzata la richiesta di iniziare il gioco (linea 692). La linea 693 realizza il loop di attesa del carattere di inizio. Ricevuta la richiesta, sono inizializzate le coordinate del cursore ai valori 0,0, in modo che esso sia posto nella prima casella in alto a sinistra (linea 710); si cancella poi il video (linea 712) e si colora la cornice in verde (linea 730). È poi inizializzata la variabile di

```
450 REM ORA COSTRUISCO DEI PERCORSI ALTERNATIVI CASUALI A FORMA DI L
460 REM I PERCORSI SONO ALMENO 10
470 I=1
480 REM CALCOLO LE COORDINATE DI UN VERTICE DELLA L
490 R1=INT(RND(R1)*25)
500 C1=INT(RND(C1)*40)
510 REM CALCOLO COORDINATE SECONDO VERTICE DELLA L
520 R2=INT(RND(R2)*25)
530 C2=INT(RND(C2)*40)
540 REM COSTRUISCO LA L
550 FOR J=R1 TO R2 STEP SGN(R2-R1)
560 LA(J,C1)=32
580 NEXT J
590 FOR J=C1 TO C2 STEP SGN(C2-C1)
610 LA(R2,J)=32
620 NEXT J
630 REM HO COSTRUITO UNA L
640 REM ORA VEDO SE HO TERMINATO
650 I=I+1
660 IF I<11 THEN GOTO 490
670 REM ORA CON PROBABILITA' DEL 70% NE COSTRUISCO ALTRI
680 IF RND(0)<.7 THEN GOTO 490
690 REM IL LABIRINTO E' COSTRUITO, CHIEDO DI INIZIARE IL GIOCO
691 POKE 214,22:PRINT
692 PRINT "PER INIZIARE DIGITA RETURN          "
693 GET A$:IF A$<>CHR$(13) THEN GOTO 693
700 REM INIZIALIZZO COORDINATE DEL GIOCATORE
710 R=0:C=0
711 REM CANCELLO IL VIDEO
712 PRINT "♥"
720 REM COLORE DELLA CORNICE IN VERDE
730 POKE VI+32,5
731 REM INIZIALIZZO IL TEMPO
732 TI$="000000"
740 REM DISEGNO GLI OTTO CARATTERI INTORNO AL GIOCATORE
750 GOSUB 10000
760 REM DISEGNO IL GIOCATORE NELLA SUA POSIZIONE
770 POKE MV+C+R*40,81
780 POKE MC+C+R*40,2
790 REM CONTROLLO SE IL GIOCATORE HA VINTO
800 IF R=24 AND C=39 THEN GOTO 2000
810 REM CONTROLLO SE E' TERMINATO IL TEMPO A DISPOSIZIONE
820 IF TI$>"000100" THEN GOTO 3000
830 REM LEGGO DIREZIONE IMPOSTATA DA TASTIERA
840 A=PEEK(197)
850 REM CALCOLO LE COORDINATE DELLA NUOVA POSIZIONE
851 NR=R:NC=C
860 IF A=41 THEN NR=R-1:GOTO 900
870 IF A=44 THEN NR=R+1:GOTO 900
880 IF A=42 THEN NC=C-1:GOTO 900
890 IF A=50 THEN NC=C+1:GOTO 900
891 REM NON E' STATO IMPARTITO ALCUN COMANDO, RIPETO L'INPUT
892 GOTO 840
899 REM CONTROLLO CHE LA NUOVA POSIZIONE SIA ACCETTABILE
900 IF NR<0 OR NR>24 THEN GOSUB 20000:GOTO 820
910 IF NC<0 OR NC>39 THEN GOSUB 20000:GOTO 820
920 IF LA(NR,NC)<>32 THEN GOSUB 20000:GOTO 820
930 REM LA NUOVA POSIZIONE E' ACCETTABILE
940 R=NR
950 C=NC
960 REM RICOMINCIA IL CICLO PRINCIPALE
970 GOTO 750
1990 REM IL GIOCATORE E' USCITO DAL LABIRINTO:CONGRATULAZIONI
2000 TM=TI
2001 REM HO MEMORIZZATO IL TEMPO IMPIEGATO
2009 FOR I=1 TO 30
2010 REM CAMBIO COLORE ALLA CORNICE ED ALLO SFONDO
2020 POKE VI+32,INT(RND(0)*16)
2030 POKE VI+33,INT(RND(0)*16)
2031 REM RITARDO
2032 FOR J=1 TO 50:NEXT J
2040 NEXT I
```

sistema TI$, che servirà per visualizzare il tempo impiegato per raggiungere l'uscita o per determinare la fine del gioco nel caso in cui sia superato il tempo massimo. La linea 750 passa il controllo alla routine 10000 che determina (linee 10000÷10070) e visualizza (linee 10090÷10150) gli otto caratteri intorno al cursore. Quindi è visualizzato il cursore (linee 770, 780) e si controlla se questo si trova nel punto di uscita del labirinto (linea 800), nel qual caso il controllo passa alla linea 2000, dove si memorizza il tempo impiegato ad uscire dal labirinto. Nelle linee successive è situato il ciclo che permette di visualizzare in rapida successione lo sfondo e la cornice del video in diversi colori determinati casualmente (linee 2009÷2030).
Si visualizza poi il tempo impiegato e si chiede se si vuole giocare nuovamente (linee 2060÷2090). La linea 2100 chiama la routine 60000 che gestisce l'input del carattere digitato dal giocatore e controlla se è un DELETE (nel qual caso cancella il carattere precedente) o se è un RETURN. Se è un RETURN il controllo torna alla linea 2110 che verifica se il giocatore ha chiesto di giocare nuovamente.
Nel caso che sia stata effettuata tale richiesta il programma torna alla linea 151 e costruisce un nuovo labirinto, altrimenti riporta lo schermo ai colori normali (linee 2140÷2170) e termina.
Nel caso che il cursore non sia arrivato all'uscita, il programma controlla se è

```
2041 REM SFONDO E CORNICE IN GRIGIO
2042 POKE VI+32,15
2043 POKE VI+33,15
2050 REM VISUALIZZO IL TEMPO E CHIEDO SE SI VUOLE GIOCARE ANCORA
2060 PRINT"♥";
2065 FOR I=1 TO 10:GET A$:NEXT I:REM CANCELLO BUFFER
2070 PRINT "QQQBRAVO, SEI USCITO DAL LABIRINTO"
2080 PRINT "QIN ";INT(TM/60+.5);" SECONDI."
2090 PRINT "QQQVUOI RIPROVARE ? [S/N] ";
2100 ZL=1:GOSUB 60000:PRINT
2110 IF IN$="S" THEN GOTO 151
2120 REM IL GIOCO E' TERMINATO
2140 POKE VI+32,14
2150 POKE VI+33,6
2160 REM CARATTERI IN GRIGIO
2170 PRINT "♥"
2180 PRINT "♥QQ"
2190 END
2980 REM IL TEMPO A DISPOSIZIONE E' TERMINATO.MANDO MESSAGGIO, VISUALIZZO IL
2990 REM LABIRINTO E CHIEDO SE SI VUOL GIOCARE ANCORA
3000 PRINT "♥QMI SPIACE, HAI IMPIEGATO TROPPO"
3010 PRINT "QTEMPO."
3020 PRINT "QQQQQQQDIGITA RRETURN■ PER ESAMINARE IL"
3021 PRINT "QLABIRINTO"
3030 GET A$:IF A$<>CHR$(13) THEN GOTO 3030
3080 REM VISUALIZZO IL LABIRINTO
3090 FOR R1=0 TO 24
3100 FOR C1=0 TO 39
3110 PO=R1*40+C1
3120 POKE MV+PO,LA(R1,C1)
3130 POKE MC+PO,5
3140 NEXT C1
3150 NEXT R1
3151 REM VISUALIZZO IL GIOCATORE
3152 POKE MV+R*40+C,81
3153 POKE MC+R*40+C,2
3160 REM RITARDO
3170 FOR I=0 TO 4000:NEXT I
3175 FOR I=1 TO 10:GET A$:NEXT I:REM CANCELLO BUFFER
3180 REM ORA CHIEDO SE SI VUOL GIOCARE ANCORA
3190 PRINT "♥QQQQ"
3200 GOTO 2090
9000 REM INIZIANO LE ROUTINES
9980 REM ROUTINE:DISEGNA GLI OTTO CARATTERI INTORNO AL GIOCATORE
10000 RP=R-1
10010 RS=R+1
10020 CP=C-1
10030 CS=C+1
10040 IF RP<0 THEN RP=0
10050 IF RS>24 THEN RS=24
10060 IF CP<0 THEN CP=0
10070 IF CS>39 THEN CS=39
10080 REM ACCENDO I PUNTI INTORNO AL GIOCATORE
10090 FOR R1=RP TO RS
10100 FOR C1=CP TO CS
10110 PO=R1*40+C1
10120 POKE MV+PO,LA(R1,C1)
10130 POKE MC+PO,5
10140 NEXT C1
10150 NEXT R1
10200 REM HO TERMINATO
10210 RETURN
19990 REM ROUTINE:EMETTE UN BEEP PER SEGNALARE UNA MOSSA ERRATA
20000 POKE SI,0
20010 POKE SI+1,50
20020 POKE SI+5,15
20030 POKE SI+6,240
20040 POKE SI+24,15
20050 POKE SI+4,33
20060 REM RITARDO
20070 FOR I=1 TO 50:NEXT I
20080 POKE SI+4,0
```

terminato il tempo a disposizione (linea 820), legge con una PEEK (linea 840) la direzione desiderata dal giocatore e, a seconda del carattere immesso, trasferisce il controllo alle diverse sezioni preposte alla gestione degli spostamenti (linee 860÷890). Queste, qualora lo spostamento richiesto porti il cursore a urtare contro un muro (linea 920) o a uscire dallo schermo (linee 900, 910), attivano la routine 20000, di generazione suoni, per avvertire il giocatore. Se invece lo spostamento richiesto è ammissibile lo si effettua (linee 940, 950) e il controllo torna al ciclo principale (linea 970), che ricomincia visualizzando gli otto caratteri intorno alla nuova posizione del cursore.
Infine, se si supera il tempo massimo previsto senza che il giocatore riesca a uscire dal labirinto, si passa alle linee 3000÷3021 che visualizzano il messaggio: «Mi spiace, hai impiegato troppo tempo. Digita RETURN per esaminare il labirinto». Nel caso in cui sia digitato RETURN sono visualizzati l'intero labirinto (linee 3090÷3150) e la posizione in cui si trovava il cursore (linee 3152, 3153), quindi è attivato un loop che permette di mantenere l'immagine sul video per un certo periodo (linea 3170). Anche in questo caso il programma domanda se si vuole giocare ancora (linea 3200).

```
20090 POKE SI+24,0
20100 RETURN
59920 REM ROUTINE:GESTISCE L'INPUT DI UNA STRINGA ALFANUMERICA
59930 REM LE VARIABILI UTILIZZATE SONO:Z6,Z7,Z8,Z9,Z8$,IN$
59940 REM IN INGRESSO VUOLE IL VALORE ZL CHE E' LA LUNGHEZZA MASSIMA DELLA
59950 REM STRINGA DA LEGGERE
59960 REM IN USCITA DA' LA STRINGA LETTA IN IN$ E LA SUA LUNGHEZZA IN Z9
59980 REM CANCELLO LA ZONA DI SCHERMO IN CUI VERRA' DIGITATA LA STRINGA
60000 FOR Z8=1 TO ZL: PRINT " ";:NEXT Z8
60010 FOR Z8=1 TO ZL: PRINT "▮";:NEXT Z8
60020 IN$="":Z7=TI
60040 REM LEGGO UN CARATTERE
60060 GET Z8$:IF Z8$<>"" THEN 60160
60080 REM ACCENSIONE E SPEGNIMENTO DEL CURSORE
60100 IF Z7<TI AND NOT(Z6) THEN PRINT "▒▮";:Z6=NOT(Z6):Z7=TI+15
60110 IF Z7<TI AND Z6 THEN PRINT " ▮";:Z6=NOT(Z6):Z7=TI+15
60120 GOTO 60060
60140 REM E' STATO DIGITATO UN CARATTERE
60160 Z8=ASC(Z8$):Z9=LEN(IN$)
60180 REM SE NON E' UN CARATTERE ALFANUMERICO, DEVE ESSERE UN RETURN O DELETE
60200 IF NOT((Z8>47 AND Z8<58) OR (Z8>64 AND Z8<91)) THEN GOTO 60320
60220 REM CONTROLLO CHE NON SIA STATA SUPERATA LA LUNGHEZZA MASSIMA
60240 IF Z9=ZL THEN GOTO 60060
60260 REM LO AGGIUNGO ALLA STRINGA IN$
60280 IN$=IN$+Z8$:PRINT Z8$;:GOTO 60060
60300 REM SE E' UN RETURN, HO TERMINATO LA LETTURA
60320 IF Z8=13 THEN PRINT " ▮";:RETURN
60340 REM SE E' DELETE, CANCELLO IN IN$ E SUL VIDEO L' ULTIMO CARATTERE DIGITATO
60360 IF Z8=20 AND Z9>0 THEN IN$=LEFT$(IN$,Z9-1):PRINT " ▮▮";:GOTO 60060
60370 GOTO 60060
60960 REM ROUTINE:INIZIALIZZAZIONE COSTANTI
60980 REM CIRCUITO VIDEO
61000 VI=53248
61020 REM CIRCUITO SUONO
61040 SI=54272
61060 REM MEMORIA VIDEO
61080 MV=1024
61100 REM MEMORIA COLORE
61120 MC=55296
61140 REM COSTANTI DI USO COMUNE
61160 ZL=9
61180 REM INIZIALIZZAZIONE CHIP SUONO
61200 FOR I=0 TO 24
61210 POKE SI+I,0
61220 NEXT I
61230 RETURN
61980 REM ROUTINE:STAMPA IL TITOLO DEL PROGRAMMA MEMORIZZATO IN PG$
62000 GOSUB 61000
62010 POKE VI+32,15
62020 POKE VI+33,15
62030 PRINT "▒▒▒▒▒▒";
62040 PRINT TAB(6);"┌────────────────────────────┐"
62050 FOR I=1 TO 5
62060 PRINT TAB(6);"|                            |"
62070 NEXT I
62080 PRINT TAB(6);"└────────────────────────────┘"
62090 PRINT TAB(6);"▒▒▒▒▒▒▒▒DIGITA ▒RETURN▒ PER PROSEGUIRE"
62100 PRINT "▒▒▒▒▒▒"
62110 FOR I=1 TO 5
62120 PRINT TAB(7);
62130 FOR J=1 TO 26
62140 PRINT "▒▒ ";
62150 NEXT J
62160 PRINT
62170 NEXT I
62190 REM ORA SCRIVO IL TITOLO
62210 PRINT "▒▒▒▒▒▒▒▒▒▒";TAB((40-LEN(PG$))/2);"▒▒";PG$
62220 GET Z9$
62230 IF Z9$<>CHR$(13) THEN GOTO 62220
62240 PRINT "▒▒";
62250 RETURN
```

Il gioco della vita

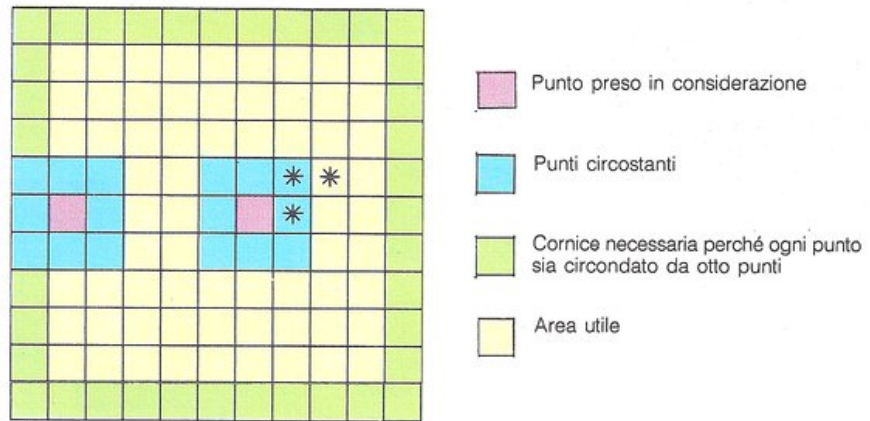
Il programma che sarà illustrato intende rappresentare, in un certo senso, le dure regole della sopravvivenza. Sappiamo infatti che un numero troppo esiguo di «individui» comporta l'estinzione di una specie per difficoltà nella riproduzione, e che un numero troppo elevato provoca una selezione naturale con l'eliminazione degli individui più deboli, ad esempio per la difficoltà di reperire cibo sufficiente per tutti. Esiste invece un punto di equilibrio che permette la sopravvivenza e la riproduzione di un numero di individui più o meno stabile. Il nostro programma simulerà questa semplice legge naturale rappresentando un «territorio» in una zona del video (pensiamo ad esempio ai territori di caccia di una tribù preistorica) e dando la possibilità al giocatore di inserirvi un certo numero di stelle (che rappresentano gli abitanti che lo popolano): a seconda del numero di stelle inserito si verrà a determinare una situazione di sovrappopolazione, di sottopopolazione o di equilibrio che sarà individuata utilizzando alcuni parametri stabiliti. Il programma, partendo dalla configurazione datagli dal giocatore, calcolerà una seconda «generazione» di stelle (cioè il numero di individui che si troverà su quel territorio dopo una generazione, in base alla legge di sopravvivenza) e tutte le generazioni successive, fino a che non sia impartito il comando che ferma l'esecuzione. Poiché la popolazione che può sopravvivere in uno stesso territorio varia in base alle condizioni sociali e tecnologiche, il lettore potrà modificare l'ampiezza del territorio e i parametri che determinano il tasso di natalità e di mortalità delle stelle (che rappresentano gli individui) apportando delle semplicissime modifiche ad alcune linee del programma.

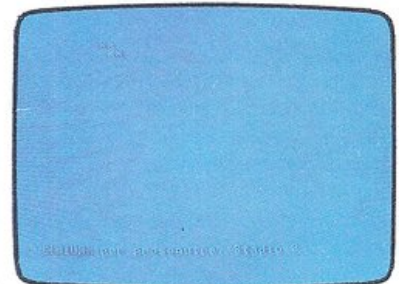
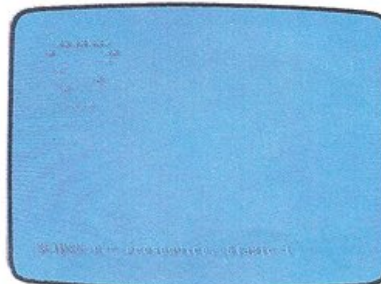
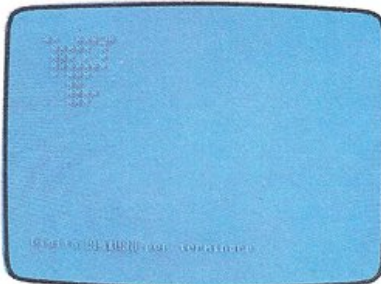
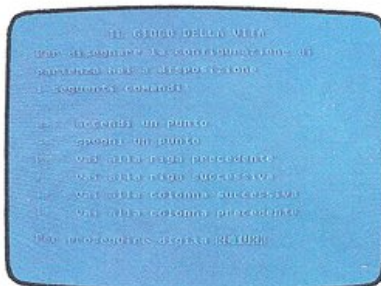
Analisi del problema

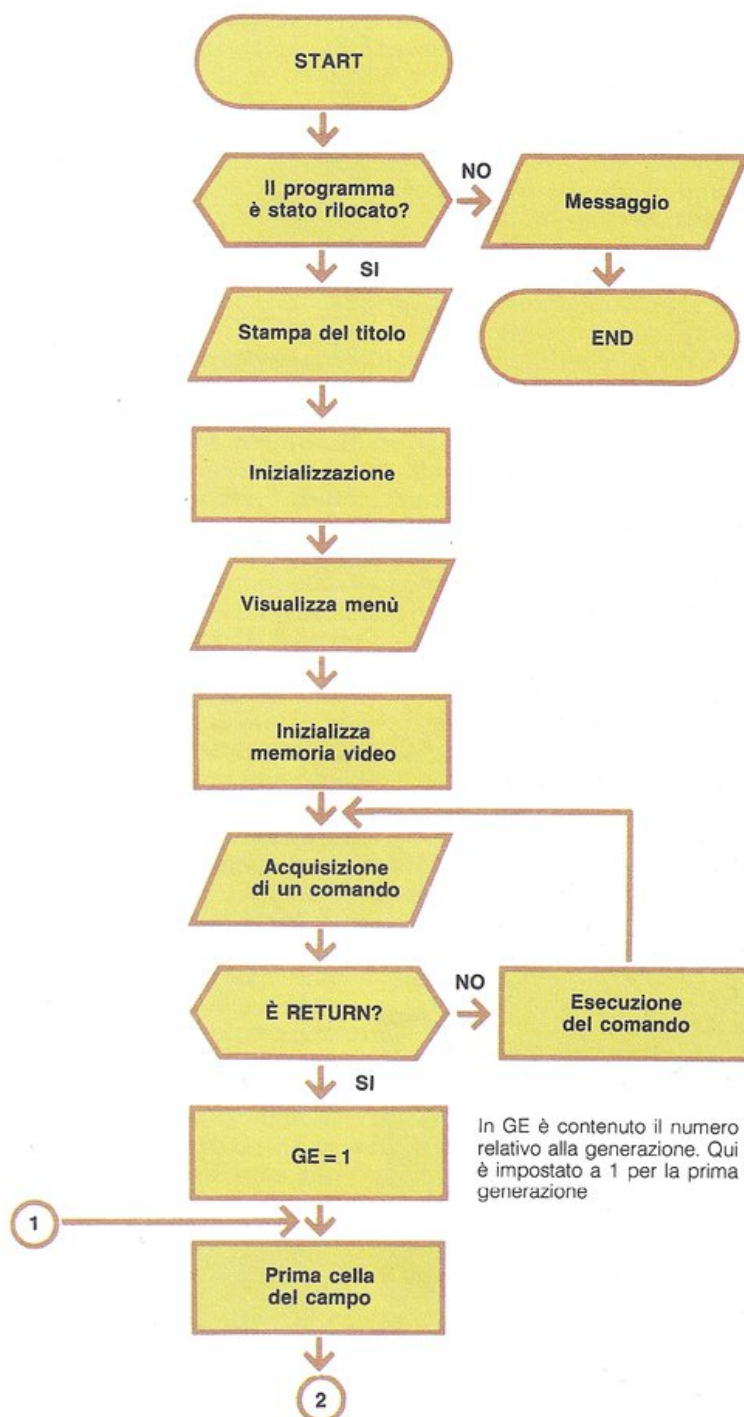
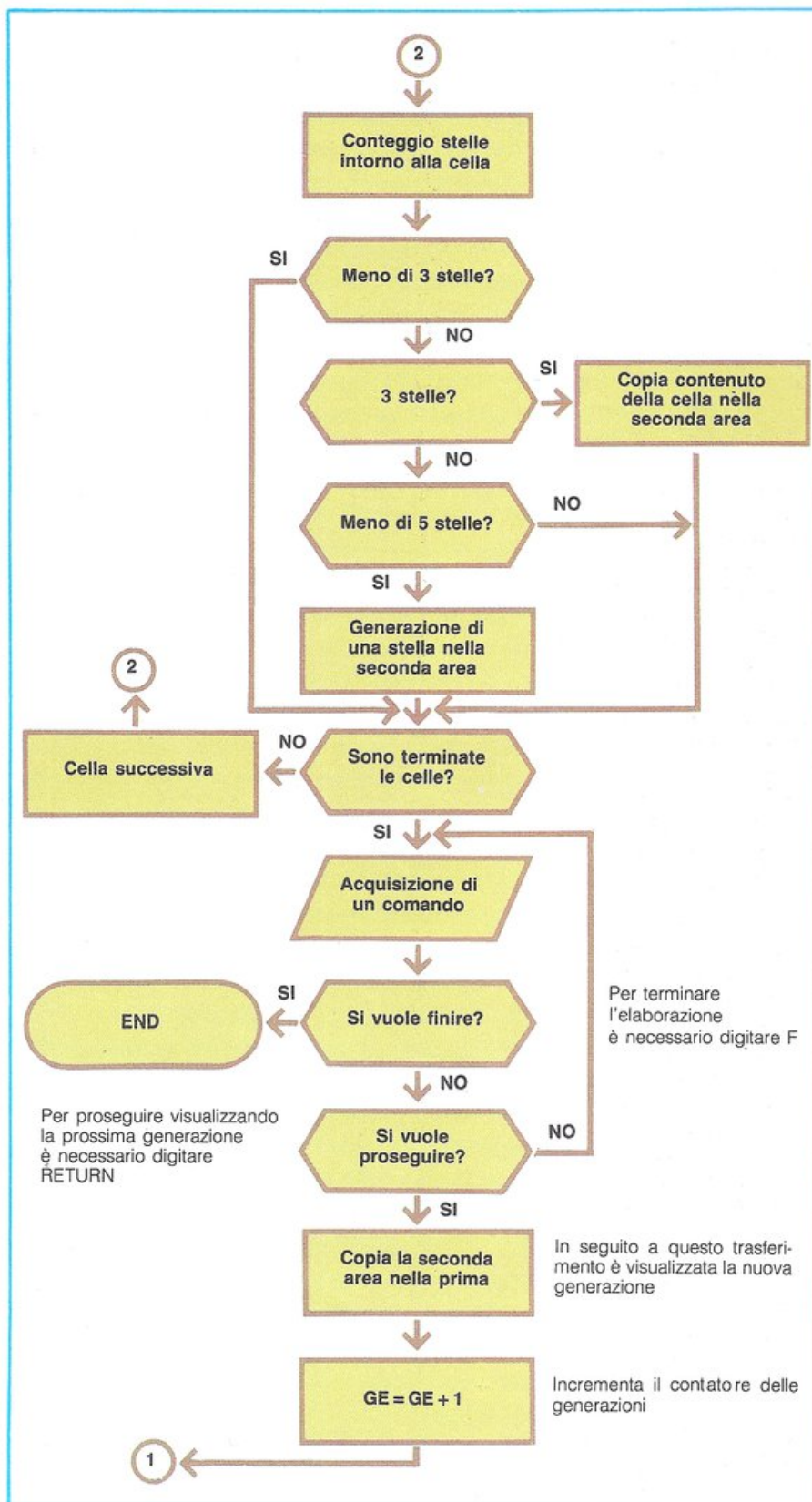
Il programma che intendiamo realizzare dovrà quindi essere in grado

- di individuare un'area di memoria dove il giocatore possa inserire la configurazione iniziale di stelle
- di gestire i comandi necessari per questo inserimento
- di calcolare una nuova generazione di stelle modificando quella iniziale in base ai parametri stabiliti
- di assumere la generazione così calcolata come nuova generazione iniziale per poter stabilire e presentare anche le successive.

La prima soluzione che viene in mente per inserire le stelle in un'area di memoria è di dimensionare una matrice. In questo caso, però, per rendere più veloce l'esecuzione del programma, individueremo direttamente un'area di memoria stabilendo l'indirizzo della prima e dell'ultima cella dell'area che si vuole riservare. Sarà scelta, in questo caso, l'area dove è collocata la memoria del video (dall'indirizzo 1024 al 2023); ciò permetterà di inserire i dati direttamente nella cella di memoria corrispondente alla posizione del cursore tramite istruzioni POKE, che vi memorizzeranno i codici dei caratteri digitati da tastiera (42 per stella e 32 per blank, cioè zona vuota). La scelta di questa particolare area di memoria consentirà inoltre la visualizzazione diretta dei caratteri che vi sono memorizzati.
Per calcolare la nuova generazione sarà necessario verificare il numero delle stelle che circondano ogni locazione della memoria del video scelta per collocarvi il territorio. Si pensi al quadretto di un quaderno e agli otto che lo circondano; il quadretto in questione corrisponde alla locazione di memoria video da analizzare, mentre gli otto circostanti rappresentano le locazioni nelle quali si deve verificare se è presente o no una

stella. L'area di memoria dove saranno memorizzate le stelle avrà la configurazione illustrata qui sotto.

Utilizzando due istruzioni cicliche FOR annidate, si verificherà il numero delle stelle che circondano ogni punto e si determinerà, in base ai parametri stabiliti, se nella nuova generazione il punto dovrà essere lasciato invariato, se vi si dovrà collocare una stella o se si dovrà eliminare la stella eventualmente presente. La nuova generazione così determinata sarà memorizzata in una seconda area di memoria di dimensioni uguali alla prima e individuata analogamente, con la sola differenza che, non trattandosi della memoria del video, i caratteri non saranno visualizzati.
A richiesta del giocatore sarà trasferita la configurazione della seconda area di memoria nella prima, e quindi sarà visualizzata la nuova generazione sullo schermo.
Sarà possibile partendo dalla seconda generazione ottenerne una terza, dalla terza una quarta e così via, fino a che il giocatore non fermi l'esecuzione del programma digitando il tasto F.
I parametri scelti per determinare le generazioni successive a quella iniziale saranno definiti come segue:

- se le stelle presenti nei punti che circondano quello in esame (che a sua volta potrà essere o una stella o un blank) sono in numero inferiore o uguale a due il punto in esame non conterrà alcuna stella nella successiva generazione
- se il punto è circondato da tre stelle il suo contenuto rimarrà invariato nella successiva generazione
- se il punto è circondato da quattro stelle conterrà a sua volta una stella nella generazione successiva
- se infine il punto è circondato da cinque o più stelle l'eventuale stella che conteneva sarà eliminata.

Diagrammi di flusso

Nella prima fase il programma esegue una verifica per controllare se la sua base è stata spostata. In caso negativo invia un messaggio contenente le istruzioni necessarie per attuare lo spostamento e termina l'esecuzione; in caso positivo stampa il titolo e inizializza le costanti. Quindi, individuate le locazioni di partenza e di arrivo delle due zone di memoria che saranno utilizzate nel seguito, si ha la stampa del menù dei comandi messi a disposizione del giocatore. Segue l'inizializzazione della zona di memoria del video, le cui celle sono riempite di blank (il codice ASCII è 32). Quindi il programma rimane in attesa di uno dei comandi previsti nel menù, ad ognuno dei quali corrisponde un movimento del cursore nelle quattro direzioni o l'accensione/spegnimento di una stella. Impostata la configurazione iniziale, per ottenere la prima generazione il giocatore deve digitare RETURN. A questo punto è spento il cursore e si inizia a calcolare il numero delle stelle da generare nella seconda area di memoria.

Questa operazione è eseguita contando i punti accesi intorno ad ogni cella; se questi sono meno di 3, nella seconda area non sarà acceso alcun punto, se sono 3 nella seconda area sarà trasferito il contenuto della cella in questione, se sono meno di 5 nella seconda area sarà generata una stella.
Si prosegue con la richiesta di un nuovo comando. Quindi, letto il comando, si controlla se è F (fine esecuzione), ed eventualmente si termina l'esecuzione, mentre se non lo è si verifica se è RETURN. In questo caso il programma, tramite l'istruzione POKE, copia la seconda area di memoria nella prima, per cui si otterrà la visualizzazione sullo schermo della generazione calcolata.
È doveroso ribadire che il campo di gioco non è null'altro che la visualizzazione della prima area di memoria.
A questo punto si incrementa GE (numero delle generazioni) per tornare a calcolare una nuova generazione di stelle.

Il programma

Il listato del programma inizia con l'elenco delle variabili utilizzate; segue l'avvertimento di spostare la base del programma se non è già stato fatto (linee 140÷210). La linea 240 affida il controllo alla routine 2070 che, inizializzate le costanti avvalendosi della routine 1920, visualizza il titolo del programma. Sono poi inizializzate le seguenti costanti:
— NR (numero righe video occupate) e NC (numero colonne video occupate) alla linea 270;
— CS e CB, in cui sono memorizzati rispettivamente il codice che individua il carattere stella e quello che rappresenta il carattere blank (linee 282 e 284);
— alcune ulteriori costanti di uso comune (linea 286). La linea 288 attribuisce i valori alle variabili RE e CE, che individuano il numero di righe e colonne effettivamente visualizzate. Sono quindi determinati gli indirizzi della prima e dell'ultima cella di ciascuna delle due zone di memoria utilizzate (linee 300, 301) ed è stabilito di presentare sul video caratteri grigi (linea 370), in minuscolo (linea 320), su sfondo blu (linee 340, 350). È il caso di notare che, avendo impostato i caratteri minuscoli, affinché il programma possa visualizzare lettere maiuscole, nella compilazione del listato è necessario digitare i tasti corrispondenti tenendo premuto contemporaneamente il tasto SHIFT. Questi

```
0 REM **********************
1 REM *IL GIOCO DELLA VITA*
2 REM **********************
4 REM VARIABILI UTILIZZATE
6 REM PG$      :MEMORIZZA IL TITOLO DEL PROGRAMMA
7 REM NR       :NUMERO DI RIGHE VIDEO OCCUPATE
8 REM NC       :NUMERO DI COLONNE VIDEO OCCUPATE
9 REM CS       :CODICE POKE DEL CARATTERE STELLA
10 REM CB      :CODICE POKE DEL CARATTERE BLANK
11 REM RE      :NUMERO EFFETTIVO DI RIGHE UTILIZZATE
12 REM CE      :NUMERO EFFETTIVO DI COLONNE UTILIZZATE
13 REM M1      :INIZIO ZONA VIDEO OCCUPATA
14 REM F1      :FINE ZONA VIDEO OCCUPATA
15 REM M2,F2   :DELIMITANO LA ZONA DI MEMORIA DOVE VIENE MEMORIZZATA
16 REM LA NUOVA GENERAZIONE
17 REM A$      :COMANDI IN INGRESSO
18 REM R,C     :PUNTATORI DI RIGA E COLONNA NELLA FASE DI EDITING
19 REM PC      :LOCAZIONE DI MEMORIA NELLA QUALE DEVE APPARIRE IL CURSORE
20 REM A       :MEMORIZZA I COMANDI NELLA FASE DI EDITING
21 REM GE      :NUMERO DI GENERAZIONI
22 REM PA      :NUMERO DI STELLE INTORNO AL PUNTO IN ESAME
23 REM PR      :VARIABILE DI COMODO
100 REM CONTROLLO CHE LA BASE DEL PROGRAMMA BASIC SIA ALLA LOCAZIONE 16384
110 IF PEEK(44)=64 THEN GOTO 240
120 REM AVVERTO CHE IL PROGRAMMA NON PUO' ANDARE IN ESECUZIONE
130 PRINT "▯▯▯▯▯▯"
140 PRINT "ATTENZIONE, HAI DIMENTICATO"
150 PRINT "▯▯DI IMMETTERE I SEGUENTI COMANDI:"
160 PRINT "▯▯▯1) POKE 44,64"
170 PRINT "▯▯2) POKE 16384,0"
180 PRINT "▯▯PRIMA DI IMMETTERLI RICORDA"
190 PRINT "▯▯DI SALVARE IL PROGRAMMA, SE QUESTO"
200 PRINT "▯NON E' GIA' PRESENTE SU NASTRO,"
210 PRINT "▯E POI DI RICARICARLO"
220 END
230 REM TITOLO ED INIZIALIZZAZIONE COSTANTI C64
240 PG$="IL GIOCO DELLA VITA"
250 GOSUB 2070
260 REM INIZIALIZZO IL NUMERO DI RIGHE E DI COLONNE DEL CAMPO DI GIOCO
270 NR=10:NC=10
280 REM INIZIALIZZAZIONE COSTANTI DI USO COMUNE
281 REM CARATTERE STELLA
282 CS=42
283 REM CARATTERE BIANCO
284 CB=32
285 REM VALORI COSTANTI
286 ZE=0:UN=1:DU=2:TR=3:QA=4:QU=40:Q1=41:T9=39:MQ=1024
287 REM NUMERO DI RIGHE E COLONNE EFFETTIVAMENTE VISUALIZZATE
288 RE=NR-1:CE=NC-1
289 REM LOCAZIONI DI PARTENZA E DI ARRIVO DELLE DUE ZONE DI MEMORIA VIDEO USATE
300 M1=1024+41:F1=M1+(RE-1)*40+CE
301 M2=2024+41:F2=M2+(RE-1)*40+CE
310 REM CARATTERI MINUSCOLI
320 PRINT CHR$(14)
330 REM SFONDO E CORNICE IN BLU
340 POKE VI+32,6
350 POKE VI+33,6
360 REM CARATTERI IN GRIGIO
370 PRINT "▯";
380 REM SPIEGAZIONE GIOCO
390 PRINT "▯          ╲L |╲┌┬ ¯¯L⊥♠ X╲|♠"
400 PRINT "▯┬ER DISEGNARE LA CONFIGURAZIONE DI "
410 PRINT "▯PARTENZA HAI A DISPOSIZIONE "
420 PRINT "▯I SEGUENTI COMANDI:"
430 PRINT "▯▯";
440 PRINT "♠:   ACCENDI UN PUNTO▯"
450 PRINT "●:   SPEGNI UN PUNTO▯"
460 PRINT "┐:   VAI ALLA RIGA PRECEDENTE▯"
470 PRINT ".:   VAI ALLA RIGA SUCCESSIVA▯"
480 PRINT ";:   VAI ALLA COLONNA SUCCESSIVA▯"
490 PRINT "L:   VAI ALLA COLONNA PRECEDENTE"
500 PRINT "▯▯┬ER PROSEGUIRE DIGITA ▯─┐ ╱─╱▯"
510 GET A$:IF A$<>CHR$(13) THEN GOTO 510
520 REM AVVISO DI ATTESA
530 PRINT"▯"
540 POKE 214,23:PRINT
550 PRINT "ATTENDERE UN MOMENTO, PREGO";
560 REM INIZIA LA FASE DI EDITING
570 REM INIZIALIZZO LA ZONA VIDEO 1024-2023
580 FOR I=1024 TO 2043
590 POKE I,CB
600 NEXT I
610 REM LEGGO LA CONFIGURAZIONE DA TASTIERA
620 GOSUB 1140
630 REM ORA NELLA ZONA 1024-2023 HO LA CONFIGURAZIONE INIZIALE
640 REM INIZIA IL PROGRAMMA
650 REM LA GENERAZIONE ATTUALE E' LA 0
660 GE=0
670 REM AVVISO DI ATTESA
680 PRINT "▯"
```

caratteri sono rappresentati, all'interno del listato, dai simboli grafici relativi alla digitazione effettuata. Come esempio si può vedere alla linea 390 la rappresentazione della scritta maiuscola IL GIOCO DELLA VITA (il cuoricino iniziale corrisponde al tasto CLR/HOME).

Le linee 390÷500 hanno il compito di visualizzare l'illustrazione dei comandi disponibili. Il loop di attesa del carattere necessario a proseguire è realizzato alla linea 510 e la visualizzazione del messaggio «Attendere un momento, prego» alla 550 (dopo essersi posizionati alla 24ª riga del video con la POKE della linea 540). È poi inizializzata la zona di memoria video, costituita dalle celle comprese tra l'indirizzo 1024 e il 2023, memorizzandovi caratteri blanks (linee 580÷600). La linea 620 passa il controllo alla routine 1140, con la quale sono inizializzate le coordinate del cursore (linee 1140, 1150) ed è stampato il messaggio «Digita RETURN per terminare» (linea 1180). Quindi il cursore è posizionato in corrispondenza della prima cella in alto a sinistra della zona di memoria effettivamente utilizzata (linea 1200) e si passa a leggere un comando digitato da tastiera (linea 1230). Se il comando è A si memorizza una stella nella cella di memoria corrispondente alla posizione del cursore (linea 1250); se è S nella medesima cella di memoria si memorizza un blank (linea 1270). Per quel che riguarda lo spostamento

```
690 POKE 214,21:PRINT"":PRINT"":PRINT""
700 PRINT "ATTENDERE UN MOMENTO, PREGO          ";
710 REM CALCOLO NUOVA GENERAZIONE
720 FOR J=M1 TO F1 STEP QU
722 FOR I=J TO J+CE-1
740 REM CONTO IL NUMERO DI PUNTI ACCESI INTORNO AL PUNTO I
760 PA=(PEEK(I-Q1)=CS)+(PEEK(I-QU)=CS)+(PEEK(I-T9)=CS)+(PEEK(I-UN)=CS)
770 PA=PA+(PEEK(I+UN)=CS)+(PEEK(I+T9)=CS)+(PEEK(I+QU)=CS)+(PEEK(I+Q1)=CS)
780 PA=-PA
810 REM CONTROLLO SE IL PUNTO RELATIVO, NELLA ZONA VIDEO M2, DEVE ESSERE ACCESO
820 REM O SPENTO, O POSTO PARI A QUELLO DELLA ZONA VIDEO M1
840 IF PA<TR THEN PR=CB:GOTO 880
850 IF PA=TR THEN PR=PEEK(I):GOTO 880
860 IF PA=QA THEN PR=CS:GOTO 880
870 PR=CB
880 POKE I+MQ,PR
890 NEXT I
900 NEXT J
970 REM ORA STAMPO RICHIESTA COMANDO E NUMERO GENERAZIONE
980 PRINT "▩"
990 POKE 214,21:PRINT"":PRINT"":PRINT""
1000 PRINT "▩▁▔▏⁄~⁄▦ PER PROSEGUIRE. ♦TADIO";GE;
1002 REM PULISCO IL BUFFER
1004 FOR I=1 TO 10:GET A$
1005 IF A$="F" THEN GOTO 1040
1006 NEXT I
1010 REM ORA LEGGO IL COMANDO
1020 GET A$:IF A$="" THEN GOTO 1020
1030 REM E' F?
1040 IF A$="F" THEN PRINT CHR$(142);"▩▩▩▩":GOSUB 1893:END
1041 REM E' <> RETURN?
1042 IF A$<>CHR$(13) THEN GOTO 1020
1043 REM E' RETURN
1044 REM COPIO LA ZONA 2 NELLA ZONA 1
1045 FOR J=M1 TO F1 STEP QU
1046 FOR I=J TO J+CE-1
1047 POKE I,PEEK(I+MQ)
1048 NEXT I
1049 NEXT J
1050 GE=GE+1
1058 REM POSSO RICOMINCIARE
1059 GOTO 680
1090 REM INIZIANO LE ROUTINES
1100 REM ROUTINE:LEGGE LA CONFIGURAZIONE DA TASTIERA
1110 REM LO SCHERMO E' GIA' RIEMPITO DI CARATTERI BIANCHI
1130 REM INIZIALIZZO POSIZIONE CURSORE
1140 R=UN
1150 C=UN
1160 REM SCRIVO UN AVVISO DALLA RIGA 23
1170 POKE 214,21:PRINT
1180 PRINT "▔IGITA ▩▁▦ PER FINIRE"
1185 PRINT "▩▔IGITA ▩▁▔▏⁄~⁄▦ PER PROSEGUIRE";
1190 REM POSIZIONO IL CURSORE
1200 PC=MV+R*QU+C
1210 POKE PC,PEEK(PC)+128
1220 REM LEGGO COMANDO
1230 A=PEEK(197)
1240 REM E' ACCENDI?
1250 IF A=10 THEN POKE PC,CS:GOTO 1210
1260 REM E' SPEGNI?
1270 IF A=13 THEN POKE PC,CB:GOTO 1210
1280 REM E' ALTO?
1290 IF A<>41 THEN GOTO 1360
1300 REM SPENGO CURSORE
1310 POKE PC,PEEK(PC)-128
1320 REM CAMBIO RIGA
1330 R=R-UN:IF R<UN THEN R=RE
1340 GOTO 1200
1350 REM E' BASSO?
1360 IF A<>44 THEN GOTO 1430
1370 REM SPENGO CURSORE
1380 POKE PC,PEEK(PC)-128
1390 REM CAMBIO RIGA
1400 R=R+UN:IF R>RE THEN R=UN
1410 GOTO 1200
1420 REM E' DESTRA?
1430 IF A<>50 THEN GOTO 1500
1440 REM SPENGO IL CURSORE
1450 POKE PC,PEEK(PC)-128
1460 REM CAMBIO COLONNA
1470 C=C+UN:IF C>CE THEN C=UN
1480 GOTO 1200
1490 REM E' SINISTRA?
1500 IF A<>CS THEN GOTO 1570
1510 REM SPENGO IL CURSORE
1520 POKE PC,PEEK(PC)-128
1530 REM CAMBIO COLONNA
1540 C=C-UN:IF C<UN THEN C=CE
1550 GOTO 1200
1560 REM E' RETURN?
```

del cursore, se il comando è P questo è spostato di una riga verso l'alto (linea 1330), se è . di una riga verso il basso (linea 1400), se è ; di una colonna verso destra (linea 1470) e se è L di una colonna verso sinistra (linea 1540). Quando si digita il tasto RETURN il controllo torna al programma principale (linee 1570÷1610) che visualizza un messaggio di attesa (linea 700) ed entra in un doppio ciclo per calcolare la nuova generazione di stelle. In questo doppio ciclo si determina il numero di stelle presenti nelle celle di memoria che circondano quella in esame (linee 760÷780) e si decide, in base ai parametri scelti, se nella cella corrispondente della seconda zona di memoria deve essere memorizzata una stella oppure un blank (linee 840÷880). Il programma esce dal ciclo dopo aver analizzato tutti i punti della configurazione iniziale e memorizzato di conseguenza la nuova generazione nella seconda area di memoria. La linea 1000 stampa poi un messaggio di richiesta di un comando, e la linea 1020 realizza il loop di attesa del comando. Se si digita F il programma termina, se invece si digita RETURN è copiata (e perciò visualizzata) la seconda area di memoria nella prima (linee 1045÷1050) e il controllo passa alla linea 680, dove ricomincia il ciclo principale. Il programma continua quindi a calcolare generazioni successive fino a quando non sia immesso il comando F.

```
1570 IF A<>1 THEN GOTO 1230
1580 REM SPENGO CURSORE
1590 POKE PC,PEEK(PC)-128
1600 REM LA ROUTINE E' TERMINATA
1610 RETURN
1620 REM ROUTINE:GESTISCE L'INPUT DI UNA STRINGA ALFANUMERICA
1630 REM LE VARIABILI UTILIZZATE SONO:Z6,Z7,Z8,Z9,Z8$,IN$
1640 REM IN INGRESSO VUOLE IL VALORE ZL CHE E' LA LUNGHEZZA MASSIMA DELLA
1650 REM STRINGA DA LEGGERE
1660 REM IN USCITA DA' LA STRINGA LETTA IN IN$ E LA SUA LUNGHEZZA IN Z9
1670 REM CANCELLO LA ZONA DI SCHERMO IN CUI VERRA' DIGITATA LA STRINGA
1680 FOR Z8=1 TO ZL: PRINT " ";:NEXT Z8
1690 FOR Z8=1 TO ZL: PRINT "█";:NEXT Z8
1700 IN$="":Z7=TI
1710 REM LEGGO UN CARATTERE
1720 GET Z8$:IF Z8$<>"" THEN 1780
1730 REM ACCENSIONE E SPEGNIMENTO DEL CURSORE
1740 IF Z7<TI AND NOT(Z6) THEN PRINT "▒█";:Z6=NOT(Z6):Z7=TI+15
1750 IF Z7<TI AND Z6 THEN PRINT " █";:Z6=NOT(Z6):Z7=TI+15
1760 GOTO 1720
1770 REM E' STATO DIGITATO UN CARATTERE
1780 Z8=ASC(Z8$):Z9=LEN(IN$)
1790 REM SE NON E' UN CARATTERE ALFANUMERICO, DEVE ESSERE UN RETURN O DELETE
1800 IF NOT((Z8>47 AND Z8<58) OR (Z8>64 AND Z8<91)) THEN GOTO 1860
1810 REM CONTROLLO CHE NON SIA STATA SUPERATA LA LUNGHEZZA MASSIMA
1820 IF Z9=ZL THEN GOTO 1720
1830 REM LO AGGIUNGO ALLA STRINGA IN$
1840 IN$=IN$+Z8$:PRINT Z8$;:GOTO 1720
1850 REM SE E' UN RETURN, HO TERMINATO LA LETTURA
1860 IF Z8=13 THEN PRINT " █";:RETURN
1870 REM SE E' DELETE, CANCELLO IN IN$ E SUL VIDEO L' ULTIMO CARATTERE DIGITATO
1880 IF Z8=20 AND Z9>0 THEN IN$=LEFT$(IN$,Z9-1):PRINT " ██";:GOTO 1720
1890 GOTO 1720
1892 REM ROUTINE:RITORNO AI COLORI NORMALI
1893 POKE 53280,14:POKE 53281,6
1894 PRINT "▒":PRINT "▒"
1895 RETURN
1900 REM ROUTINE:INIZIALIZZAZIONE COSTANTI
1910 REM CIRCUITO VIDEO
1920 VI=53248
1930 REM CIRCUITO SUONO
1940 SI=54272
1950 REM MEMORIA VIDEO
1960 MV=1024
1970 REM MEMORIA COLORE
1980 MC=55296
1990 REM COSTANTI DI USO COMUNE
2000 ZL=9
2010 REM INIZIALIZZAZIONE CHIP SUONO
2020 FOR I=0 TO 24
2030 POKE SI+I,0
2040 NEXT I
2050 RETURN
2060 REM ROUTINE:STAMPA IL TITOLO DEL PROGRAMMA MEMORIZZATO IN PG$
2070 GOSUB 1920
2080 POKE VI+32,15
2090 POKE VI+33,15
2100 PRINT "▒▒▒▒▒";
2110 PRINT TAB(6);" ┌──────────────────────────────┐"
2120 FOR I=1 TO 5
2130 PRINT TAB(6);" │                              │"
2140 NEXT I
2150 PRINT TAB(6);" └──────────────────────────────┘"
2160 PRINT TAB(6);"▒▒▒▒▒▒▒DIGITA ▒RETURN▒ PER PROSEGUIRE"
2170 PRINT "▒▒▒▒▒"
2180 FOR I=1 TO 5
2190 PRINT TAB(7);
2200 FOR J=1 TO 26
2210 PRINT "▒▒ ";
2220 NEXT J
2230 PRINT
2240 NEXT I
2250 REM ORA SCRIVO IL TITOLO
2260 PRINT "▒▒▒▒▒▒▒▒▒";TAB((40-LEN(PG$))/2);"▒▒";PG$
2270 GET Z9$
2280 IF Z9$<>CHR$(13) THEN GOTO 2270
2290 PRINT "▒▒";
2300 RETURN
```

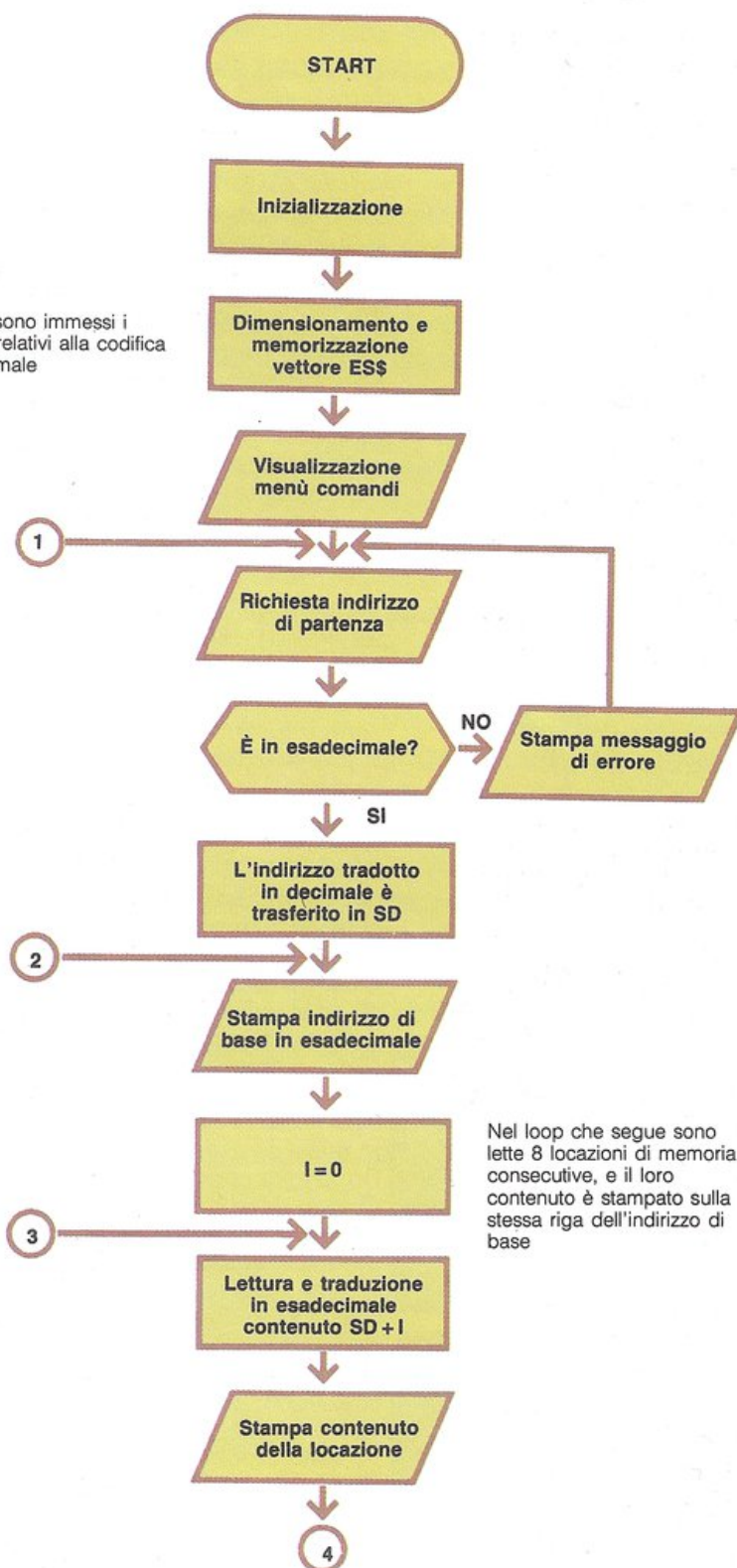
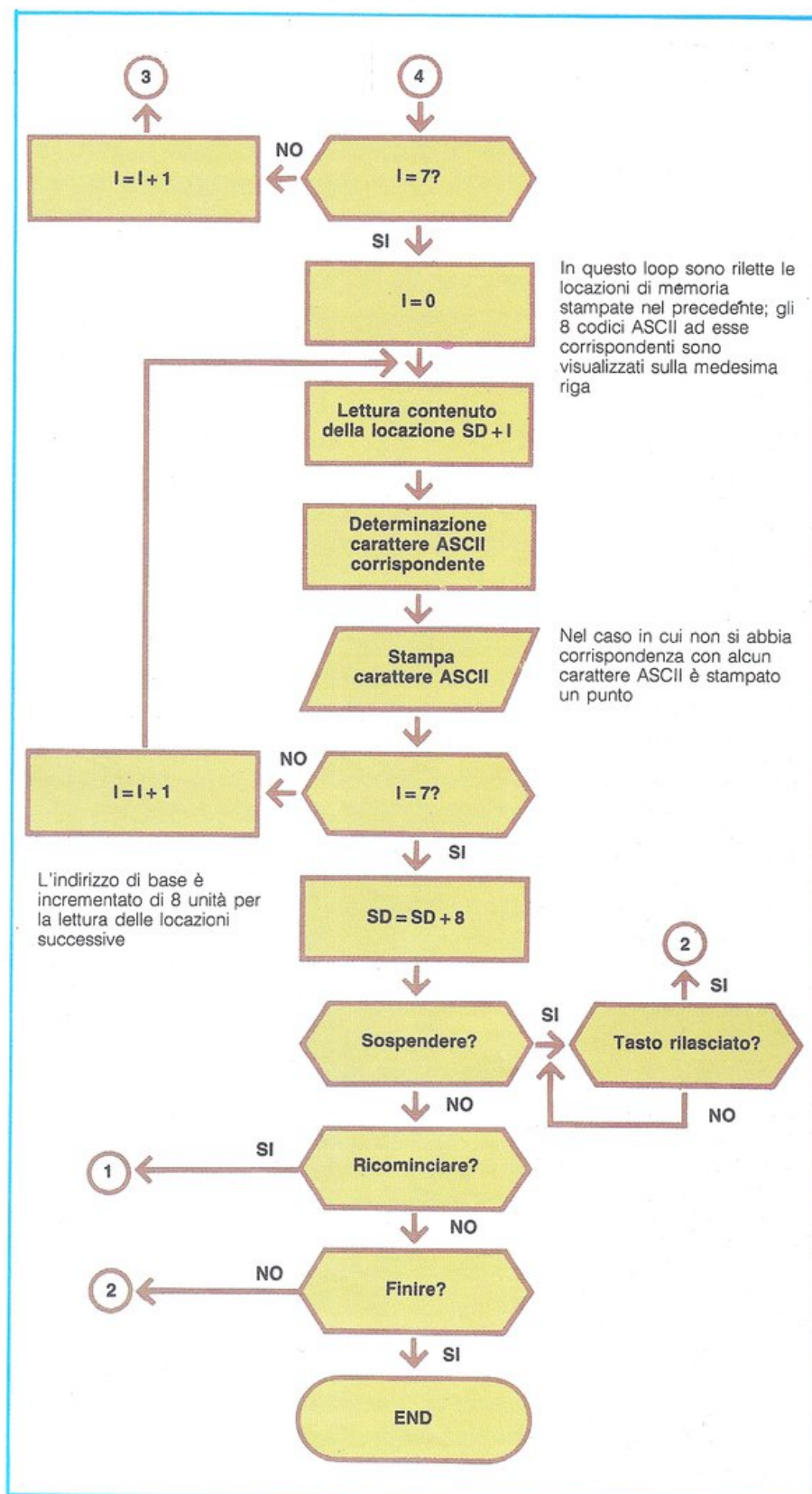
Memory dump

Consideriamo ora la realizzazione di un programma che permetta di effettuare un «dump» della memoria. Il significato letterale del verbo inglese «to dump» è «scaricare», il che ci dà già un'idea di cosa vogliamo che faccia il nostro programma. Infatti, quando si esegue un dump da un calcolatore, sono fornite tutte le configurazioni di cifre binarie presenti in ogni indirizzo di memoria (quindi, in un certo senso, si scarica il contenuto delle celle di memoria). Generalmente il calcolatore fornisce, oltre alle configurazioni binarie, anche la loro traduzione in numeri ottali ed esadecimali, e l'eventuale carattere ASCII corrispondente. L'ASCII (American Standard Code for Information Interchange) è appunto un codice che permette di rappresentare i caratteri.
Usualmente ogni codice è a lunghezza fissa, cioè ogni configurazione è caratterizzata dallo stesso numero di bits. In particolare l'ASCII è un codice ad otto bits e permette quindi di rappresentare $2^8=256$ caratteri, a differenza dei codici a 6 bits (BCD, FIELDATA, ecc.), che ne possono rappresentare solo $2^6=64$. Se in seguito alla lettura di una locazione di memoria è riportato un carattere, non è detto che in quella cella sia memorizzato proprio il codice ASCII di quel carattere. Infatti si potrebbe verificare l'eventualità che la configurazione di cifre binarie di quella cella di memoria sia uguale a quella del codice ASCII di un carattere, pur avendo un altro significato.

Analisi del problema

Generalmente il dump è effettuato sui grandi calcolatori e i dati relativi sono registrati su memoria di massa. Noi invece vogliamo vedere come si può progettare un programma che effettui il dump sul C-64 e ne visualizzi i dati sullo schermo. Per motivi di velocità di esecuzione e di spazio, limiteremo le informazioni fornite alla traduzione esadecimale delle configurazioni binarie contenute nelle celle di memoria e agli eventuali caratteri ASCII corrispondenti. Il nostro programma dovrà quindi fornire, per un qualsiasi indirizzo di memoria dei 65536 disponibili sul C-64, la configurazione di cifre binarie in esso presente (tradotta in esadecimale) e il carattere eventualmente codificato da tale configurazione. Inoltre il programma potrà fornire, in esadecimale, anche i codici operativi, gli operandi, i dati, gli indirizzi, ecc., in cui sono state tradotte le sue stesse istruzioni Basic (a partire dalla locazione 2048, 800 in esadecimale).
Il programma dovrà chiedere, per prima cosa, un indirizzo di partenza in esadecimale. Infatti, poiché si fa riferimento per comodità agli indirizzi di memoria usando la notazione esadecimale, prevedremo, anche in questo caso, che i dati siano accettati e stampati in questa notazione. Dopo aver controllato la correttezza dei simboli usati per introdurre l'indirizzo di partenza, sarà necessario convertire la notazione esadecimale in quella decimale per poter leggere con una PEEK il contenuto della cella di memoria specificata. Per motivi di spazio saranno stampati su una stessa riga i contenuti di otto celle di memoria successive, evidenziando in alto la sequenza degli indirizzi cui si fa riferimento. Successivamente sarà necessario controllare se la configurazione binaria di ogni cella di memoria corrisponde ad un carattere ASCII stampabile. Se è così il carattere corrispondente sarà determinato (tramite la funzione CHR$) e visualizzato, altrimenti sarà inserito un punto nella porzione di video dove sono riportati i caratteri. Infine, prima di iniziare a decodificare in un'altra riga i successivi otto indirizzi di memoria, si dovrà verificare se è stato digitato uno dei comandi previsti: S per sospendere momentaneamente la stampa, R per interrompere l'esecuzione del programma e fornire un nuovo indirizzo di partenza, ed F per determinare la fine dell'esecuzione del programma.

Diagrammi di flusso

*Il primo blocco del diagramma di flusso si occupa dell'inizializzazione delle costanti e della stampa del titolo.
Nel secondo blocco si dimensiona il vettore ES$ e vi si memorizzano i simboli della codifica esadecimale, che saranno utilizzati per eseguire le conversioni fra i sistemi di numerazione (esadecimale-decimale).
Dopo aver stampato il menù dei comandi, il programma passa a visualizzare un messaggio nel quale è richiesto all'utente da quale locazione di memoria deve iniziare il dump.
Successivamente il programma legge il valore immesso tramite la tastiera, e prosegue verificando se tale valore è esadecimale.
In caso di errore si stampa il relativo messaggio e si chiede nuovamente l'indirizzo di partenza. Si esegue poi la trasformazione della stringa, espressa in esadecimale, nel corrispondente valore decimale, che è trasferito nella variabile SD.
È quindi stampato l'indirizzo di base (SD tradotto in esadecimale) e successivamente si entra in un ciclo per leggere e stampare gli otto bytes contenuti nelle locazioni di memoria a partire da quella puntata da SD. Il programma prosegue entrando nel ciclo in cui è determinato e stampato il carattere ASCII, relativo ad ogni contenuto letto nel loop precedente.*

Il primo blocco che si incontra in questo nuovo ciclo è quello dove avviene la lettura del byte della locazione, attraverso la funzione PEEK. Poi si passa alla trasformazione del contenuto in carattere ASCII con la funzione CHR$, e lo si stampa. Anche le istruzioni contenute nei blocchi di questo ciclo sono ripetute per otto volte. Per calcolare la successiva locazione di base si incrementa la variabile di indirizzo SD di otto bytes. Infine, per consentire all'utente di sospendere momentaneamente l'esecuzione, di andare ad un nuovo indirizzo o di uscire dal programma, sono inseriti tre blocchi di verifica. A ciascuno di essi è associato un comando che permette di realizzare l'operazione scelta. Se non è digitato alcun comando valido (S, R o F) l'elaborazione continua con l'esame delle 8 locazioni di memoria successive.

Il programma

Il programma inizia con la linea 120, dove è memorizzato il titolo nella stringa PG$, che sarà utilizzata poi nella routine 62000 chiamata alla linea successiva. A tale routine è affidata la presentazione del titolo e l'inizializzazione delle costanti (linee 60960÷62250). È successivamente inizializzato il vettore ES$, usato per le conversioni esadecimale-decimale e decimale-esadecimale (linee 180÷210). Le linee 231÷238 servono per stampare il menù dei comandi. Alla linea 250 si effettua l'inizializzazione della variabile CL, che servirà per determinare ogni quante linee di dump si ristamperanno le informazioni relative ai dati contenuti in ogni colonna che sarà visualizzata sullo schermo. Il controllo passa poi alla routine 10000 che, chiesto all'utente l'indirizzo di memoria da cui desidera iniziare il dump, chiama la routine 60000. Questa legge il carattere immesso con una GET (linea 60060), determina il lampeggiamento del cursore (linee 60100, 60110) e verifica se il carattere introdotto è alfanumerico, RETURN o DELETE (linea 60200). Nel caso si tratti di un DELETE cancella l'ultimo carattere digitato e ne attende uno nuovo (linea 60360); se invece si tratta di un carattere alfanumerico, dopo aver verificato che non si sia superata la lunghezza massima prevista

```
0 REM **************
1 REM *MEMORY DUMP*
2 REM **************
4 REM VARIABILI UTILIZZATE
6 REM I,J         :CONTATORI DI CICLO
7 REM CL          :CONTA LE LINEE STAMPATE, FINO A 20
8 REM C$          :VARIABILE DI COMODO
9 REM SD          :INDIRIZZO DELLA LOCAZIONE DI PARTENZA DEL DUMP
10 REM DI         :USATA NELLA CONVERSIONE ESADECIMALE-DECIMALE
11 REM CN         :VALORE DECIMALE DI INGRESSO PER LA ROUTINE DI CONVERSIONE
12 REM CN$        :STRINGA ESADECIMALE DI USCITA PER LA ROUTINE DI CONVERSIONE
13 REM RE         :USATA NELLA CONVERSIONE DECIMALE-ESADECIMALE
14 REM C          :VARIABILE DI COMODO
15 REM A$         :SERVE PER SVUOTARE IL BUFFER DI INGRESSO
16 REM A          :USATA PER LEGGERE IL COMANDO DA TASTIERA
17 REM ES$(15)    :USATO NELLA CONVERSIONE DECIMALE-ESADECIMALE
110 REM TITOLO ED INIZIALIZZAZIONE COSTANTI C64
120 PG$="MEMORY DUMP"
130 GOSUB 62000
140 REM VIENE ESEGUITO UN DUMP DELLA MEMORIA NEL FORMATO ESADECIMALE ED ASCII
150 REM OGNI VENTI RIGHE DI STAMPA, VIENE STAMPATO UN SEPARATORE
160 REM COSTRUISCO IL VETTORE USATO PER LA CONVERSIONE DECIMALE-ESADECIMALE
180 DIM ES$(15)
190 FOR I=0 TO 15
200 READ ES$(I)
210 NEXT I
220 REM CARATTERI IN NERO
230 PRINT "■";
231 REM VISUALIZZO IL MENU' DEI COMANDI"
232 PRINT"■■SE VUOI SOSPENDERE TIENI PREMUTO S"
234 PRINT"■■SE VUOI RICOMINCIARE PREMI R"
236 PRINT"■■SE VUOI FINIRE PREMI F"
238 PRINT"■■"
240 REM CONTALINEE
250 CL=0
260 REM CHIEDO LA LOCAZIONE DI PARTENZA, IN ESADECIMALE
270 GOSUB 10000
280 REM IN SD C'E' LA LOCAZIONE DI PARTENZA IN DECIMALE
290 REM STAMPO SEPARATORE DI QUADRO, SE CL E' 0
300 IF CL=0 THEN PRINT "■BASE  00 01 02 03 04 05 06 07     ASCII":PRINT
310 REM CONVERTO L'INDIRIZZO IN ESADECIMALE
320 CN=SD
330 GOSUB 11000
340 PRINT CN$;
350 REM ORA STAMPO I CONTENUTI DELLE 8 LOCAZIONI DA CN A CN+7
360 FOR I=0 TO 7
370 CN=PEEK(SD+I)
380 GOSUB 11000
390 PRINT TAB(6+3*I);RIGHT$(CN$,2);
400 NEXT I
410 REM HO STAMPATO I CONTENUTI IN ESADECIMALE
420 REM ORA STAMPO I VALORI IN ASCII
430 FOR I=0 TO 7
440 C=PEEK(SD+I)
450 REM CONTROLLO SE E' UN CARATTERE STAMPABILE O NO
460 C$="."
470 IF (C>31 AND C<96) THEN C$=CHR$(C)
480 REM STAMPO IL CODICE ASCII
490 PRINT TAB(31+I);C$;
500 NEXT I
510 REM VADO A CAPO
520 PRINT
530 REM INCREMENTO INDIRIZZO
540 SD=SD+8
550 REM PRIMA DI INIZIARE LA PROSSIMA LINEA, CONTROLLO SE C'E' UN COMANDO
```

di quattro caratteri, lo memorizza nella stringa IN$ (linea 60280) e ne attende uno nuovo. Qualora infine venga digitato un RETURN (linea 60320) è restituito il controllo alla routine 10000, che verifica se effettivamente l'indirizzo di memoria desiderato è stato introdotto in notazione esadecimale (linee 10090÷10130). In caso negativo è stampato un messaggio di avvertimento e si chiede nuovamente di introdurre un indirizzo di partenza (linee 10110÷10130), altrimenti si convertono i caratteri introdotti in notazione decimale (linee 10150÷10220) e si restituisce il controllo alla linea 300.
È quindi controllata la variabile CL e, se questa ha valore zero, sono stampate le informazioni relative ad ogni colonna visualizzata (linea 300). Si converte poi l'indirizzo di memoria in esadecimale chiamando la routine 11000, e con un'istruzione ciclica FOR (linee 360÷400) sono lette (linea 370), convertite in esadecimale (linea 380) e stampate (linea 390) le configurazioni binarie contenute in otto indirizzi di memoria adiacenti. Con un'ulteriore istruzione FOR (linee 430÷500) sono letti nuovamente i contenuti delle 8 celle di memoria (linea 440), si controlla se codificano caratteri ASCII stampabili (linea 470) e si visualizza il carattere corrispondente (linea 490). Nel caso in cui la rappresentazione ASCII del byte non esista è stampato un punto. Si determina quindi l'indirizzo della successiva cella di memoria da prendere in esame (linea

```
560 REM PULISCO IL BUFFER DI INPUT
570 FOR I=1 TO 10:GET A$:NEXT I
580 A=PEEK(197)
590 REM SE E' 'S' SOSPENDO TEMPORANEAMENTE LA STAMPA
600 IF A=13 THEN GOTO 580
610 REM SE E' 'F' TERMINA IL PROGRAMMA
620 IF A=21 THEN PRINT "▒▒▒":GOSUB 11110:END
630 REM SE E' 'R' CHIEDO IL NUOVO INDIRIZZO DI DUMP
640 IF A=17 THEN GOTO 250
650 REM INCREMENTO IL CONTALINEE
660 CL=CL+1:IF CL=20 THEN CL=0
670 GOTO 300
1100 CN$=""
9990 REM ROUTINE:CHIEDO LA LOCAZIONE DI PARTENZA, IN ESADECIMALE
10000 PRINT "▒▒LOCAZIONE DI PARTENZA,"
10010 PRINT "▒IN ESADECIMALE ? ";
10020 REM L'INDIRIZZO E' AL PIU' DI 4 CIFRE ESADECIMALI
10030 ZL=4
10040 GOSUB 60000:PRINT
10050 REM CONTROLLO CHE L'INDIRIZZO SIA CORRETTO
10060 IF IN$="" THEN SD=0:RETURN
10070 I=1
10080 C$=MID$(IN$,I,1)
10090 IF (C$>="0" AND C$<="9") OR (C$>="A" AND C$<="F") THEN GOTO 10131
10100 REM LA STRINGA CONTIENE UN CARATTERE DIVERSO DA 0..9,A..F
10110 PRINT "▒ATTENZIONE, UN NUMERO ESADECIMALE"
10120 PRINT "▒NON PUO' CONTENERE IL CARATTERE ";C$
10130 GOTO 10000
10131 I=I+1
10132 IF I<=LEN(IN$) THEN GOTO 10080
10140 REM CONVERTO IN DECIMALE
10141 SD=0
10150 FOR I=LEN(IN$) TO 1 STEP -1
10160 REM CALCOLO IL VALORE DECIMALE DEL DIGIT ESADECIMALE IN ESAME
10170 C$=MID$(IN$,I,1)
10180 DI=ASC(C$)-ASC("0")
10190 IF C$>="A" THEN DI=ASC(C$)-ASC("A")+10
10200 REM IN DI C'E' UN NUMERO TRA 0 E 15
10210 SD=SD+DI*16↑(LEN(IN$)-I)
10220 NEXT I
10230 REM HO TERMINATO LA CONVERSIONE
10240 RETURN
10980 REM ROUTINE:CONVERTE LA CIFRA DECIMALE CONTENUTA IN CN, NELLA STRINGA
10990 REM ESADECIMALE IN CN$
11000 CN$=""
11010 IF CN=0 THEN GOTO 11070
11020 RE=CN-INT(CN/16)*16
11030 CN=INT(CN/16)
11040 CN$=ES$(RE)+CN$
11050 GOTO 11010
11060 REM LA STRINGA CN$ DEVE ESSERE LUNGA 4 CARATTERI
11070 IF LEN(CN$)<4 THEN CN$="0"+CN$:GOTO 11070
11080 REM CONVERSIONE TERMINATA
11090 RETURN
11100 REM ROUTINE:RITORNO AI COLORI NORMALI
11110 POKE53280,14:POKE53281,6
11120 PRINT"▒"
11125 PRINT"▒"
11130 RETURN
49990 REM CIFRE ESADECIMALI
50000 DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
53281 ,5
59920 REM ROUTINE:GESTISCE L'INPUT DI UNA STRINGA ALFANUMERICA
59930 REM LE VARIABILI UTILIZZATE SONO:Z6,Z7,Z8,Z9,Z8$,IN$
59940 REM IN INGRESSO VUOLE IL VALORE ZL CHE E' LA LUNGHEZZA MASSIMA DELLA
```

540) ed infine si controlla se è stato digitato un comando. Se si mantiene premuto il tasto S la stampa è sospesa (linee 580÷600) fino a che il tasto non sia rilasciato. Qualora si digiti F il programma termina (linea 620). Premendo il tasto R invece il programma chiede un nuovo indirizzo di memoria da cui iniziare il dump (linea 640). Se non è digitato alcun comando valido si passa a incrementare e a controllare la variabile CL (linea 660), per visualizzare poi il contenuto delle 8 celle di memoria successive (linea 670).

```
59950 REM STRINGA DA LEGGERE
59960 REM IN USCITA DA' LA STRINGA LETTA IN IN$ E LA SUA LUNGHEZZA IN Z9
59980 REM CANCELLO LA ZONA DI SCHERMO IN CUI VERRA' DIGITATA LA STRINGA
60000 FOR Z8=1 TO ZL: PRINT " ";:NEXT Z8
60010 FOR Z8=1 TO ZL: PRINT "█";:NEXT Z8
60020 IN$="":Z7=TI
60040 REM LEGGO UN CARATTERE
60060 GET Z8$:IF Z8$<>"" THEN 60160
60080 REM ACCENSIONE E SPEGNIMENTO DEL CURSORE
60100 IF Z7<TI AND NOT(Z6) THEN PRINT "██";:Z6=NOT(Z6):Z7=TI+15
60110 IF Z7<TI AND Z6 THEN PRINT " █";:Z6=NOT(Z6):Z7=TI+15
60120 GOTO 60060
60140 REM E' STATO DIGITATO UN CARATTERE
60160 Z8=ASC(Z8$):Z9=LEN(IN$)
60180 REM SE NON E' UN CARATTERE ALFANUMERICO, DEVE ESSERE UN RETURN O DELETE
60200 IF NOT((Z8>47 AND Z8<58) OR (Z8>64 AND Z8<91)) THEN GOTO 60320
60220 REM CONTROLLO CHE NON SIA STATA SUPERATA LA LUNGHEZZA MASSIMA
60240 IF Z9=ZL THEN GOTO 60060
60260 REM LO AGGIUNGO ALLA STRINGA IN$
60280 IN$=IN$+Z8$:PRINT Z8$;:GOTO 60060
60300 REM SE E' UN RETURN, HO TERMINATO LA LETTURA
60320 IF Z8=13 THEN PRINT " █";:RETURN
60340 REM SE E' DELETE, CANCELLO IN IN$ E SUL VIDEO L' ULTIMO CARATTERE DIGITATO
60360 IF Z8=20 AND Z9>0 THEN IN$=LEFT$(IN$,Z9-1):PRINT " ██";:GOTO 60060
60370 GOTO 60060
60960 REM ROUTINE:INIZIALIZZAZIONE COSTANTI
60980 REM CIRCUITO VIDEO
61000 VI=53248
61020 REM CIRCUITO SUONO
61040 SI=54272
61060 REM MEMORIA VIDEO
61080 MV=1024
61100 REM MEMORIA COLORE
61120 MC=55296
61140 REM COSTANTI DI USO COMUNE
61160 ZL=9
61180 REM INIZIALIZZAZIONE CHIP SUONO
61200 FOR I=0 TO 24
61210 POKE SI+I,0
61220 NEXT I
61230 RETURN
61980 REM ROUTINE:STAMPA IL TITOLO DEL PROGRAMMA MEMORIZZATO IN PG$
62000 GOSUB 61000
62010 POKE VI+32,15
62020 POKE VI+33,15
62030 PRINT "██████";
62040 PRINT TAB(6);"┌──────────────────────────┐"
62050 FOR I=1 TO 5
62060 PRINT TAB(6);"│                          │"
62070 NEXT I
62080 PRINT TAB(6);"└──────────────────────────┘"
62090 PRINT TAB(6);"██████████DIGITA █RETURN█ PER PROSEGUIRE"
62100 PRINT "██████"
62110 FOR I=1 TO 5
62120 PRINT TAB(7);
62130 FOR J=1 TO 26
62140 PRINT "██ ";
62150 NEXT J
62160 PRINT
62170 NEXT I
62190 REM ORA SCRIVO IL TITOLO
62210 PRINT "██████████";TAB((40-LEN(PG$))/2);"██";PG$
62220 GET Z9$
62230 IF Z9$<>CHR$(13) THEN GOTO 62220
62240 PRINT "██";
62250 RETURN
```

Archivio

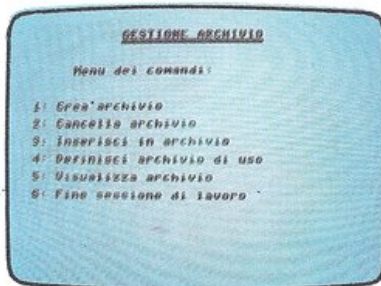
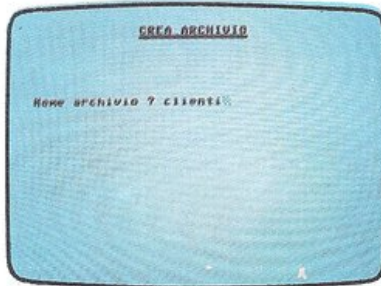
L'utilità, se non la necessità, di disporre di grandi quantità di informazioni è sempre più evidente in molti campi dell'attività umana. Si pensi agli uffici di immatricolazione di vetture, agli uffici elettorali, al fisco, alle prefetture, alle imprese, all'anagrafe. Tutti questi uffici, e molti altri, si trovano a dover gestire grandissime quantità di dati e di nominativi, e hanno quindi enormi problemi di catalogazione e di conservazione. Per questi motivi si va diffondendo sempre più la gestione dei dati con l'ausilio del calcolatore. I computer offrono infatti la possibilità di memorizzare le informazioni su supporti magnetici (nastri, dischi, ecc.) che, a parità di ingombro, possono contenere un numero di dati molto più elevato rispetto ai supporti cartacei. Vengono offerti anche altri vantaggi: le informazioni così memorizzate possono essere lette, modificate e cancellate molto più velocemente, utilizzando appositi programmi che svolgono queste funzioni. Inoltre, la conservazione dei dati si può paragonare a quella di un archivio tradizionale: un file di dati (schedario) è diviso in records (schede) a loro volta divisi in campi (le informazioni contenute in una scheda: cognome, nome...). La gestione dei dati è invece effettuata mediante programmi (sostituiscono gli impiegati che una volta dovevano cercare le schede, inserirle, modificarle, ecc.) che leggono i dati memorizzati sui supporti magnetici, li caricano nella memoria centrale e qui eseguono le operazioni che eventualmente sono state richieste. Inoltre poiché i dischi, i nastri, ecc. possono essere conservati per periodi di tempo molto lunghi, presentano, da questo punto di vista, le stesse caratteristiche dei supporti cartacei, e in più possono essere cancellati e riutilizzati.
Il programma che presentiamo vuol essere un esempio di come si possa gestire un archivio mediante un calcolatore.

Analisi del problema

All'atto di memorizzare le informazioni su supporto magnetico è possibile definire diversi tipi di files, che si distinguono tra di loro per struttura e modalità di accesso. In particolare, sull'unità a disco del CBM-64 si possono creare

- files sequenziali, che hanno struttura ed accesso sequenziale, il cui difetto principale è legato proprio a questa loro natura, in quanto devono essere ogni volta letti dall'inizio alla fine
- files random, o files ad accesso diretto, che permettono di leggere ogni singolo record del file senza dover leggere ogni volta l'intero file. Ogni record occupa necessariamente un blocco del dischetto, cui si può accedere utilizzando i numeri di traccia e di settore che lo individuano
- «relative files», anch'essi ad accesso diretto, che permettono di dividere i files in records sceglendone anche la lunghezza; a sua volta ogni record può essere diviso in campi delle dimensioni volute. Non è quindi necessario che un record abbia lunghezza uguale ad un blocco perché sono previsti dei «puntatori» che individuano l'inizio di ciascun record.

Poiché il relative file offre una maggiore flessibilità d'uso e la possibilità di dividere i records in campi, sarà conveniente utilizzare questo tipo di file per la creazione e la gestione di un archivio: sarà così possibile utilizzare un campo per ogni informazione desiderata (cognome, nome e telefono in questo caso). La scelta di un file ad accesso casuale (come un «relative file») impedisce l'utilizzo di un'unità a nastro, in quanto questa permette solo la gestione sequenziale dei dati.
Stabilito il tipo di file più adatto alla memorizzazione di una certa cate-

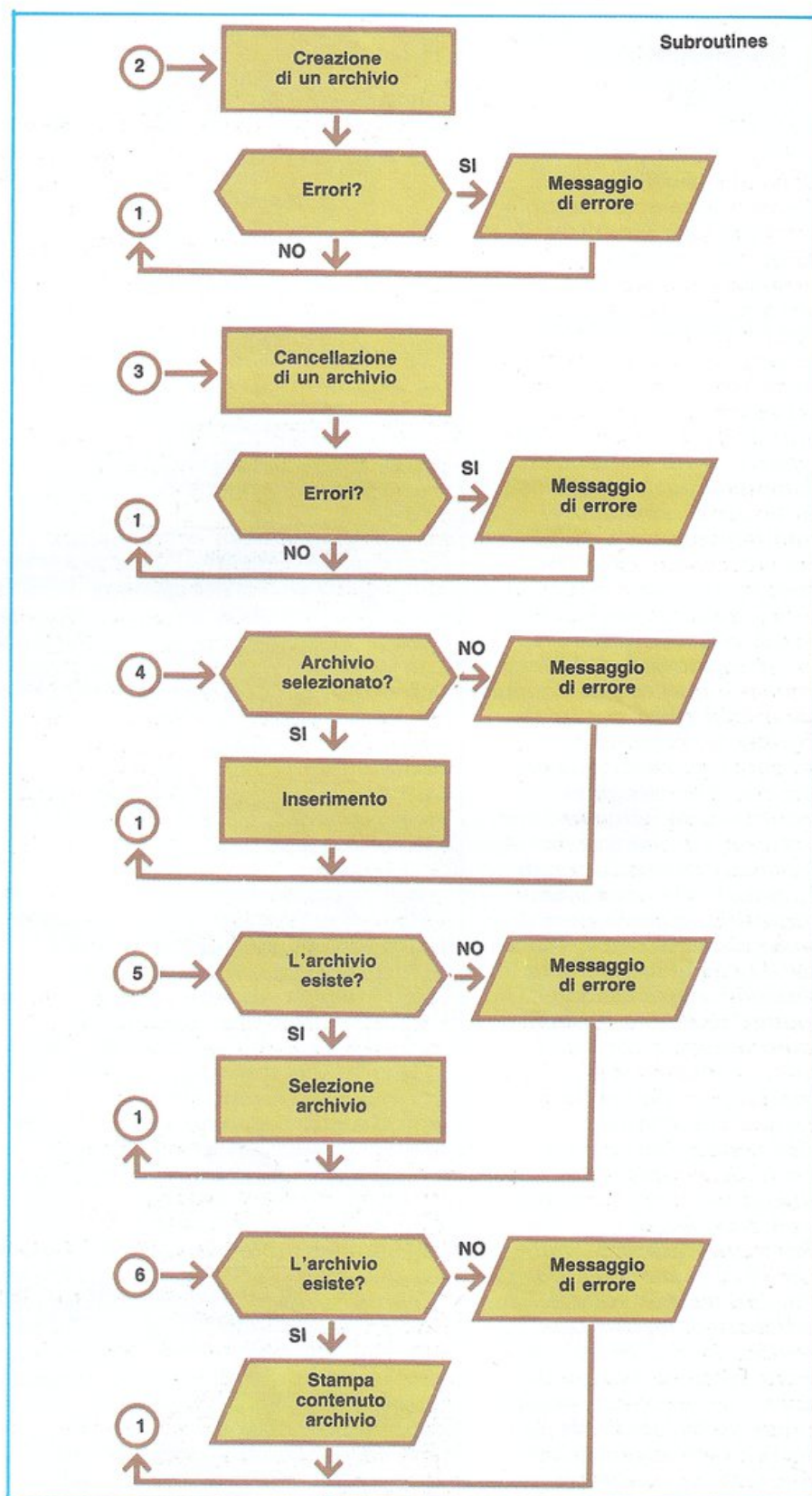
goria di dati, analizziamo quali sono le operazioni che dovrebbe essere in grado di svolgere un programma di gestione di un archivio:

- creazione dei files che compongono l'archivio
- scelta del file su cui si vuole operare
- inserimento dei dati in un record
- modifica dei dati di un record
- cancellazione di un record
- stampa di un singolo record
- stampa di un intero file
- cancellazione di un intero file.

Per semplificare la sua struttura, il nostro programma dovrà invece essere in grado di svolgere solo le seguenti operazioni:

- creazione dei files
- scelta del file su cui si vuole operare
- inserimento dei dati in un record
- visualizzazione dei records di un file
- cancellazione di un intero file.

La creazione di un file è un'operazione molto semplice; infatti il programma dovrà solo aggiungere il nome scelto dall'utente per individuare il file nell'apposita istruzione OPEN che serve a crearlo sul disco. Poiché si vuole creare un «relative file», l'istruzione OPEN avrà la configurazione che permette di stabilire la lunghezza dei records del file. Sarà necessario inoltre memorizzare, nel primo record, un valore che permetterà di individuare il primo record disponibile per effettuare un'inserzione. Quando si vorrà operare su di un file il programma dovrà controllare che questo sia stato precedentemente creato e informarne l'utente. Scelto così un file esistente si potranno effettuare due operazioni: inserimento e visualizzazione. Per poter inserire delle informazioni, il programma dovrà innanzi tutto leggere nel primo record del file il valore che individua il record disponibile per l'inserzione, per potersi così posizionare sul primo campo del record libero (per inserirvi il cognome), sul secondo campo (per inserirvi il nome) e sul terzo (per il numero di telefono). Finito l'inserimento dei dati, dovrà incrementare di una unità il valore che individua il nuovo record libero per le successive inserzioni. Quando si vorranno invece visualizzare dati memorizzati in precedenza, il programma dovrà stampare il nome del file richiestogli dall'utente, controllare che non sia vuoto e visualizzare il primo record occupato; saranno poi stampati tutti i successivi records fino al primo trovato vuoto. Per cancellare un file sarà sufficiente fornire il nome del file che si vuole eliminare, e il programma invierà i necessari comandi al disco. Dovrà anche essere prevista la visualizzazione del codice e del tipo di errore verificatosi, nel caso in cui durante lo svolgimento di una di queste operazioni avvenga un errore di lettura o scrittura sul disco. Di fondamentale importanza sarà aprire e chiudere tutti i canali utilizzati per comunicare con i files, oltre ad aprire e chiudere il canale di comando (il numero 15) necessario per trasmettere le istruzioni contenute nel programma all'unità disco.
Come sarà possibile notare leggendo il listato, questo programma è in realtà l'unione di sottoprogrammi, ognuno dei quali svolge un determinato compito (creazione, cancellazione, ecc.); perciò risulterà facile aggiungervi la possibilità di gestire altre operazioni; basterà infatti scrivere un programma che svolga una particolare operazione (ad esempio la ricerca di un determinato record individuato da un codice o dal cognome memorizzatovi) e inserirlo, con gli opportuni collegamenti, come un ulteriore sottoprogramma del programma «archivio».

Diagrammi di flusso

Dopo aver inizializzato le costanti e stampato il titolo il programma visualizza il menù dei comandi.
Il menù visualizzato contiene tutti i comandi che permettono all'utente di gestire l'archivio.
Nella fase successiva, il programma, dopo aver acquisito un comando dalla tastiera, entra nel ciclo che determina quale esecuzione associare al comando. Il primo blocco che si incontra controlla se il comando impartito è la creazione di un archivio. Se è così viene richiesto il nome da attribuire al file, si inizializza il primo record del file e si verifica se l'operazione ha avuto successo; qualora l'operazione non abbia avuto successo il programma visualizza un messaggio d'errore. Con il test successivo si verifica se il comando impone la cancellazione di un archivio. In caso positivo si richiede quale archivio debba essere cancellato, quindi si elimina il file in cui è registrato. Si passa ora a verificare se il comando è quello relativo all'inserzione di un elemento; qualora lo sia, si controlla se l'archivio richiesto è stato selezionato (aperto), dopo di che sono letti e memorizzati i dati che l'utente immette. Dopo l'inserzione è la volta della selezione di un archivio. A questo punto il programma, dopo aver verificato se è questo il comando prescelto, controlla che il file sia stato creato, e quindi passa a definirlo e ad aprirlo

logicamente. Il blocco di verifica successivo determina se il comando è l'invio in stampa di un archivio; ciò avviene dopo aver controllato se l'archivio è stato selezionato (aperto). Nell'ultimo blocco si controlla se il comando è quello di uscita; qualora lo sia, si termina l'esecuzione, e in caso contrario si ritorna alla visualizzazione del menù.
In questa versione il programma è dedicato alla gestione di un'agenda telefonica; i campi dei records sono infatti denominati: cognome, nome e telefono. Le possibilità di applicazione sono naturalmente molto più vaste, e questo può essere evidenziato cambiando i nominativi precedenti nei più generici: campo 1, campo 2 e campo 3. Utilizzando ad esempio il programma per la memorizzazione dei risultati delle partite di calcio effettuate durante il campionato, si possono identificare i files con l'indicazione della giornata di campionato (p. es., settima giornata) e in ciascun record riportare: nel campo 1 il nome della squadra di casa (p. es., Juventus), nel campo 2 il nome della squadra ospite (p. es., Roma) e nel campo 3 il risultato (p. es., 1-1).
Va notato che nel trasferimento dei campi al disco questi ultimi sono trattati come stringhe. Poiché alcuni caratteri (in particolare , e :) hanno per il driver significato di comandi, è necessario evitare di utilizzare questi caratteri in fase di digitazione dei dati, a meno di non prevedere alcuni controlli che saranno illustrati nel programma «Editor» (pag. 177).

Il programma

Dopo la stampa del titolo e l'inizializzazione delle costanti (linee 110÷122), alla linea 124 è aperto il canale di comando e alla 140 è stabilito di presentare i caratteri in minuscolo e in nero. Troviamo poi le istruzioni di visualizzazione del menù comandi (linee 140÷250) e la chiamata alla routine 10000 che gestisce l'immissione dei comandi da tastiera. Come si è già chiarito nella descrizione del programma «Il gioco della vita», anche in questo caso i caratteri grafici che compaiono in alcune linee (ad esempio nella 160) rappresentano lettere maiuscole. Quando è digitato un comando previsto, il controllo torna alla linea 290 che, a seconda della funzione richiesta, chiama la routine demandata a realizzarla. La routine che crea un file è la 20000; essa chiede il nome che si vuole dare al file da creare (linea 20030) e passa il controllo alla routine 60000, che gestisce la memorizzazione dei caratteri digitati da tastiera. Qualora non sia immesso nessun carattere, è ripresentato il menù comandi (linea 20060), altrimenti viene creato il file (linea 20100). Nell'istruzione OPEN che serve alla creazione del file sono presenti, dopo l'indicazione del numero del file, del numero di individuazione dell'unità disco e del canale, il nome da noi scelto per il file (in IN$) e l'indicazione della lunghezza di ogni record

```
0 REM **********
1 REM *ARCHIVIO*
2 REM **********
4 REM VARIABILI UTILIZZATE
6 REM I,I9    :CONTATORI DI CICLO
7 REM LA$     :MEMORIZZA IL NOME DELL'ARCHIVIO CORRENTE
8 REM IN$,A$  :INPUT DA TASTIERA
9 REM E       :CODICE NUM. DELL'ERRORE EFFETTUATO NEL TRASFERIRE I DATI SU DISCO
10 REM E$     :DENOMINAZIONE DEL TIPO DI ERRORE EFFETTUATO
11 REM T,S    :TRACCIA E SETTORE DEL BLOCCO IN CUI SI E' VERIFICATO L'ERRORE
12 REM BL     :INDIRIZZO DI UN RECORD DEL FILE LA$
13 REM BL$    :STR$ DI BL
14 REM RL     :PARTE BASSA DELL'INDIRIZZO DEL BLOCCO BL
15 REM RH     :PARTE ALTA DELL'INDIRIZZO DEL BLOCCO BL
100 REM TITOLO ED INIZIALIZZAZIONE COSTANTI C64
110 PG$="A R C H I V I O"
120 GOSUB 62000
121 REM INIZIALIZZAZIONE COSTANTI
122 LA$=""
123 REM APRO IL CANALE PER TRASMETTERE I COMANDI AL DRIVER
124 OPEN 15,8,15
130 REM CARATTERI IN NERO E MINUSCOLI
140 PRINT "▯■";CHR$(14)
150 REM STAMPO MENU COMANDI
160 PRINT "▯";TAB(11);"|‾▯|◝Γ/‾ ▯_‾├╳Γ"
170 PRINT TAB(11);"▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬"
180 PRINT "▯▯     \ENU DEI COMANDI:"
190 PRINT "▯▯▯1: ⌐REA ARCHIVIO"
200 PRINT "▯2: ⌐ANCELLA ARCHIVIO"
210 PRINT "▯3: ◝NSERISCI IN ARCHIVIO"
230 PRINT "▯4: ‾EFINISCI ARCHIVIO DI USO"
240 PRINT "▯5: XISUALIZZA ARCHIVIO"
250 PRINT "▯6: _INE SESSIONE DI LAVORO"
260 REM LEGGO IL COMANDO DA TASTIERA
270 GOSUB 10000
280 REM ESEGUO LA ROUTINE RELATIVA AL COMANDO DATO
290 ON VAL(A$) GOSUB 20000,21000,22000,23000,24000,25000
300 REM RITORNO AL MENU DEI COMANDI
310 GOTO 140
9990 REM ROUTINE:LEGGE UN COMANDO DA TASTIERA E CONTROLLA SE E' CORRETTO
10000 FOR I=1 TO 10:GET A$:NEXT I
10010 REM HO SVUOTATO IL BUFFER DI INGRESSO
10020 REM LEGGO COMANDO
10030 GET A$:IF A$="" THEN GOTO 10030
10040 REM CONTROLLO CHE IL COMANDO SIA CORRETTO
10050 IF (A$>="1") AND (A$<="6") THEN RETURN
10060 REM EMETTO UN BEEP E LEGGO IL PROSSIMO COMANDO
10070 POKE SI+24,15
10080 POKE SI,0
10090 POKE SI+1,40
10100 POKE SI+5,15
10110 POKE SI+6,240
10120 POKE SI+4,17
10130 FOR I=1 TO 50:NEXT I
10140 REM TERMINA IL BEEP
10150 POKE SI+24,0
10160 POKE SI+4,0
10170 GOTO 10000
10990 REM ROUTINE:STAMPA IL CODICE D'ERRORE DEL DISCO
11000 POKE 214,15:PRINT
11010 PRINT "▯TTENZIONE, ERRORE DEL DRIVER"
11020 PRINT "▯/UMERO :";E
11030 PRINT "TIPO   :";E$
11040 PRINT "BLOCCO :";T;S
11050 REM INIZIALIZZO IL DRIVER
11060 PRINT#15,"I"
11070 REM CHIEDO DI CONTINUARE
11080 PRINT "▯▯_‾| ⌐_/▯ PER CONTINUARE"
11090 GET IN$:IF IN$<>CHR$(13) THEN GOTO 11090
11100 RETURN
19990 REM ROUTINE:CREA ARCHIVIO
20000 PRINT "▯";TAB(13);"⌐_‾▯ ▯_⌐├╳◝Γ"
20010 PRINT TAB(13);"▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬"
20020 REM LEGGO IL NOME DELL'ARCHIVIO DA CREARE
20030 PRINT "▯▯▯▯▯▯▯/OME ARCHIVIO ? ";
20040 ZL=10:GOSUB 60000:PRINT
20050 REM SE NON E' STATA DIGITATA UNA STRINGA:RETURN
20060 IF IN$="" THEN RETURN
20090 REM APRO IL FILE IN$
20100 OPEN 4,8,4,IN$+",L,"+CHR$(48)
20110 REM LEGGO STATUS
20120 INPUT#15,E,E$,T,S
20130 REM CONTROLLO ERRORI
20140 IF E<20 THEN GOTO 20250
20150 REM C'E STATO UN ERRORE, NE STAMPO IL CODICE
20160 GOSUB 11000
20170 REM CHIUDO I CANALI APERTI
20180 CLOSE 4
20210 RETURN
20240 REM INIZIALIZZO IL BLOCCO 1
```

[CHR$(48)]. Si controlla poi (linea 20120), tramite il canale di comando, se l'unità a disco si è venuta a trovare in stato di errore e, qualora ciò sia avvenuto, è visualizzato il messaggio che indica il tipo di errore (routine 11000) e chiuso il canale utilizzato per il tentativo di creazione del file (linea 20180). Se non si sono verificati errori il programma si posiziona all'inizio del 1° campo del primo record del file appena creato (linea 20250), e vi memorizza il valore che servirà ad individuare il primo record libero per un'inserzione (linea 20260). L'istruzione della linea 20250 è detta «comando di posizione», e individua il file, l'indirizzo del record (suddiviso in parte bassa e parte alta) e la posizione all'interno del record. In questo caso viene individuato il file n° 4 [CHR$(4)], il suo primo record [CHR$(1)CHR$(0)] e ci si posiziona al primo carattere [CHR$(1)] del record individuato. Chiuso poi il canale utilizzato (linea 20290), la routine termina (linea 20310), ed è ripresentato il menù comandi.

La routine 21000, che cancella un file, chiede il nome del file da eliminare (linea 21050) e controlla che il nome sia effettivamente immesso (linea 21080); quindi elimina il file sul disco (linea 21100) e controlla se la cancellazione è avvenuta regolarmente (linee 21110÷21130).

La routine 22000 gestisce l'inserzione delle informazioni. Alla linea 22009 si controlla se è stato scelto un file in cui operare l'inserimento ed eventualmente è

```
20250 PRINT#15,"P"CHR$(4)CHR$(1)CHR$(0)CHR$(1)
20260 PRINT#4,"2"
20270 REM HO CREATO ED INIZIALIZZATO IL FILE RANDOM
20280 REM TERMINA ROUTINE
20290 CLOSE 4
20310 RETURN
20990 REM ROUTINE:CANCELLA ARCHIVIO
21000 PRINT "♫";TAB(11);"¬♠/⌐⅂⅃♠ ♠_¬├╳╲Γ"
21010 PRINT TAB(11);"▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔"
21040 REM CHIEDO NOME ARCHIVIO DA ELIMINARE
21050 PRINT "▓▓▓▓/OME ARCHIVIO ? ";
21060 ZL=10:GOSUB 60000:PRINT
21070 REM CONTROLLO CHE LA STRINGA SIA STATA DIGITATA
21080 IF IN$="" THEN RETURN
21081 IF IN$=LA$ THEN LA$="":REM ELIMINO L'ARCHIVIO DI LAVORO
21090 REM ELIMINO IL FILE
21100 PRINT#15,"S0:"+IN$
21110 INPUT#15,E,E$,T,S
21120 REM SE E=1 ALLORA TUTTO A POSTO
21130 IF E=1 THEN RETURN
21140 REM IL FILE NON E' STATO ELIMINATO PER QUALCHE MOTIVO
21150 REM VISUALIZZO ERRORE E RETURN
21160 GOSUB 11000
21220 RETURN
21980 REM ROUTINE:INSERISCE UN ELEMENTO NELL'ARCHIVIO DI LAVORO,
21990 REM SE E' STATO DEFINITO
22000 PRINT "♫";TAB(7);"INSERZIONE DI UN ELEMENTO"
22001 PRINT TAB(7);"▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔"
22009 IF LA$<>"" THEN GOTO 22040
22010 PRINT "▓▓▓▓♠TTENZIONE, L'ARCHIVIO DI LAVORO"
22020 PRINT "▓NON E' STATO DEFINITO."
22021 PRINT "▓╲NSERZIONE IMPOSSIBILE."
22022 PRINT "▓▓▓S_⅂| ⁄_/▓ PER PROSEGUIRE"
22023 GET IN$:IF IN$<>CHR$(13) THEN GOTO 22023
22024 RETURN
22040 OPEN 4,8,4,LA$
22050 REM LEGGO L'INDIRIZZO DEL PROSSIMO BLOCCO IN CUI SCRIVERE
22060 REM MI POSIZIONO ALL'INIZIO DEL BLOCCO 1
22070 PRINT#15,"P"CHR$(4)CHR$(1)CHR$(0)CHR$(1)
22080 INPUT#4,BL$
22090 BL=VAL(BL$)
22100 REM CALCOLO PARTE ALTA E PARTE BASSA DELL'INDIRIZZO DEL BLOCCO IN CUI
22110 REM EFFETTUARE L'INSERZIONE
22120 RL=BL-INT(BL/256)*256
22130 RH=INT(BL/256)
22140 REM MI POSIZIONO SUL BLOCCO BL-ESIMO
22150 PRINT#15,"P"CHR$(4)CHR$(RL)CHR$(RH)CHR$(1)
22160 REM LEGGO LE STRINGHE DA MEMORIZZARE
22170 REM MI POSIZIONO SULLA RIGA 10
22180 POKE 214,9:PRINT
22190 REM LEGGO IL COGNOME
22200 PRINT "¬OGNOME ? ";TAB(15);
22210 ZL=15:GOSUB 60000:PRINT
22220 IF IN$="" THEN GOTO 22180
22230 REM MEMORIZZO IL COGNOME
22231 PRINT#15,"P"CHR$(4)CHR$(RL)CHR$(RH)CHR$(1)
22240 PRINT#4,IN$
22250 REM MI POSIZIONO AL 17-MO POSTO
22260 PRINT#15,"P"CHR$(4)CHR$(RL)CHR$(RH)CHR$(17)
22270 REM LEGGO IL NOME
22280 POKE 214,11:PRINT
22300 PRINT "/OME  ? ";TAB(15);
22310 ZL=15:GOSUB 60000:PRINT
22320 IF IN$="" THEN GOTO 22280
22330 REM MEMORIZZO IL NOME
22340 PRINT#4,IN$
22350 REM MI POSIZIONO AL 33-MO POSTO
22360 PRINT#15,"P"CHR$(4)CHR$(RL)CHR$(RH)CHR$(33)
22370 REM LEGGO IL NUMERO DI TELEFONO
22380 POKE 214,13:PRINT
22400 PRINT "|ELEFONO ? ";TAB(15);
22410 ZL=15:GOSUB 60000
22420 IF IN$="" THEN GOTO 22380
22430 REM MEMORIZZO IL NUMERO DI TELEFONO
22440 PRINT#4,IN$
22450 REM HO TERMINATO L'INPUT
22460 REM MEMORIZZO NEL BLOCCO 1 L'INDIRIZZO DEL PROSSIMO BLOCCO LIBERO
22470 BL=BL+1
22490 REM MI POSIZIONO ALL'INIZIO DEL BLOCCO 1
22500 PRINT#15,"P"CHR$(4)CHR$(1)CHR$(0)CHR$(1)
22510 PRINT#4,STR$(BL)
22530 REM CHIUDO I CANALI APERTI
22540 CLOSE 4
22560 RETURN
22990 REM ROUTINE:DEFINISCE L'ARCHIVIO DI LAVORO
23000 PRINT "♫";TAB(5);"▔⅂_╮/╲♦╲Γ/⌐ ♠_¬├╳╲Γ ⌐╲ L♠╳Γ_Γ"
23010 PRINT TAB(5);"▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔"
23020 PRINT "▓▓▓▓▓/OME ARCHIVIO ? ";
23030 ZL=10:GOSUB 60000:PRINT
23040 IF IN$="" THEN RETURN
```

visualizzato un messaggio per avvertire che non è possibile effettuare inserzioni senza prima definire il file, ed è ripresentato il menù comandi (linee 22010÷22024). È poi aperto il file scelto (linea 22040) e si legge il contenuto del primo record (linee 22070, 22080), che serve a determinare il record libero dove poter effettuare l'inserimento. Alla linea 22090 è trasformata la stringa letta nel primo record in valore numerico, per poter calcolare l'indirizzo del record libero (linee 22120, 22130). Ci si posiziona poi sul primo campo del record disponibile per l'inserimento (linea 22150) e si richiede il cognome da memorizzare (linea 22200). L'immissione dei caratteri che compongono il cognome è gestita dalla routine 60000, che restituisce una stringa la quale viene memorizzata nel primo campo (linee 22231, 22240). Con l'istruzione della linea 22260 ci si posiziona all'inizio del 2° campo (dal 17° al 32° carattere) prima di chiedere e leggere il nome (linee 21280÷22320), per la successiva memorizzazione (linea 22340). È poi memorizzato il numero di telefono nel 3° campo del record in modo analogo (linee 22360÷22440). Completato quindi l'inserimento dei dati, è incrementato il valore che determina il primo record libero (linea 22470), e il risultato è memorizzato nel primo record del file (linee 22500, 22510) per poter effettuare la prossima inserzione. Infine si chiude il canale utilizzato e si

```
23050 REM CONTROLLO L'ESISTENZA DELL'ARCHIVIO
23070 OPEN 4,8,4,IN$
23080 INPUT#15,E,E$,T,S
23090 IF E<20 THEN LA$=IN$:CLOSE 4:RETURN
23100 REM C'E' UN ERRORE
23110 GOSUB 11000
23120 CLOSE 4
23140 RETURN
23990 REM ROUTINE:VISUALIZZA ARCHIVIO DI LAVORO, SE E' STATO DEFINITO
24000 PRINT "";TAB(12);""
24001 PRINT TAB(12);"────────────"
24002 REM CONTROLLO CHE L'ARCHIVIO DI LAVORO SIA STATO DEFINITO
24003 IF LA$<>"" THEN GOTO 24010
24004 PRINT "ATTENZIONE, L'ARCHIVIO DI LAVORO"
24005 PRINT "NON E' STATO DEFINITO."
24006 PRINT " PER PROSEGUIRE"
24007 GET IN$:IF IN$<>CHR$(13) THEN GOTO 24007
24008 RETURN
24009 REM STAMPO IL NOME DELL'ARCHIVIO DI LAVORO
24010 PRINT "-ONTENUTO DELL'ARCHIVIO ";LA$;""
24019 OPEN 4,8,4,LA$
24020 REM MI POSIZIONO SUL BLOCCO 1
24030 PRINT#15,"P"CHR$(4)CHR$(1)CHR$(0)CHR$(1)
24040 REM LEGGO L'INDIRIZZO DEL PRIMO BLOCCO LIBERO
24050 INPUT#4,BL$
24060 BL=VAL(BL$)
24100 REM CONTROLLO SE IL FILE E' VUOTO
24110 IF BL>2 THEN GOTO 24180
24120 REM IL FILE E' VUOTO
24130 PRINT "L'ARCHIVIO ";LA$;" E' VUOTO."
24140 PRINT " PER PROSEGUIRE."
24150 GET IN$:IF IN$<>CHR$(13) THEN GOTO 24150
24151 CLOSE 4
24160 RETURN
24170 REM L'ARCHIVIO NON E' VUOTO,LO STAMPO
24180 FOR I=2 TO BL-1
24190 REM MI POSIZIONO SUL BLOCCO I-ESIMO
24200 RL=I-INT(I/256)*256
24210 RH=INT(I/256)
24220 PRINT#15,"P"CHR$(4)CHR$(RL)CHR$(RH)CHR$(1)
24230 REM LEGGO E STAMPO COGNOME
24240 POKE 214,14:PRINT
24250 PRINT "-OGNOME";TAB(15);
24260 INPUT#4,IN$
24270 PRINT IN$
24280 PRINT "OME";TAB(15);
24290 REM MI POSIZIONO SUL 17-MO CARATTERE
24300 PRINT#15,"P"CHR$(4)CHR$(RL)CHR$(RH)CHR$(17)
24310 INPUT#4,IN$
24320 PRINT IN$
24330 PRINT "ELEFONO";TAB(15);
24340 REM MI POSIZIONO SUL 33-MO CARATTERE
24350 PRINT#15,"P"CHR$(4)CHR$(RL)CHR$(RH)CHR$(33)
24360 INPUT#4,IN$
24370 PRINT IN$
24380 PRINT " PER CONTINUARE"
24390 GET IN$:IF IN$<>CHR$(13) THEN GOTO 24390
24400 REM CANCELLO LE LINEE DALLA 15 ALLA 24
24410 POKE 214,14:PRINT
24420 FOR I9=1 TO 9
24430 PRINT "                                        "
24440 NEXT I9
24450 REM STAMPO, SE C'E', IL PROSSIMO RECORD
24460 NEXT I
24470 REM HO TERMINATO LA STAMPA DELL'ARCHIVIO LA$
24480 POKE 214,14:PRINT
24490 PRINT "L' ARCHIVIO E' TERMINATO."
24500 PRINT " PER PROSEGUIRE."
24510 GET IN$:IF IN$<>CHR$(13) THEN GOTO 24510
24520 REM CHIUDO I CANALI APERTI
24530 CLOSE 4
24550 RETURN
24990 REM ROUTINE:TERMINA LA SESSIONE DI LAVORO
25000 PRINT "";CHR$(142)
25010 CLOSE 15
25011 POKE53281,6:POKE53280,14
25012 PRINT""
25013 PRINT""
25020 END
59920 REM ROUTINE:GESTISCE L'INPUT DI UNA STRINGA ALFANUMERICA
59930 REM LE VARIABILI UTILIZZATE SONO:Z6,Z7,Z8,Z9,Z8$,IN$
59940 REM IN INGRESSO VUOLE IL VALORE ZL CHE E' LA LUNGHEZZA MASSIMA DELLA
59950 REM STRINGA DA LEGGERE
59960 REM IN USCITA DA' LA STRINGA LETTA IN IN$ E LA SUA LUNGHEZZA IN Z9
59980 REM CANCELLO LA ZONA DI SCHERMO IN CUI VERRA' DIGITATA LA STRINGA
60000 FOR Z8=1 TO ZL: PRINT " ";:NEXT Z8
60010 FOR Z8=1 TO ZL: PRINT "";:NEXT Z8
60020 IN$="":Z7=TI
60040 REM LEGGO UN CARATTERE
60060 GET Z8$:IF Z8$<>"" THEN 60160
```

ripresenta il menù dei comandi.
La scelta del file su cui si vuole operare è gestita dalla routine 23000 che, chiesto come al solito il nome del file (linea 23020), lo apre e ne memorizza il nome nella stringa LA$ (linee 23070÷23090), che sarà poi usata nella routine di inserimento dati.
La routine 24000 gestisce la visualizzazione del contenuto dei records di un file. Per prima cosa controlla se è stato definito il file di lavoro ed eventualmente visualizza il messaggio di avvertimento (linee 24001÷24008).
La linea 24010 stampa poi il nome del file e la 24019 lo apre. È quindi letto l'indirizzo del primo record libero (linea 24050) per poter controllare se il file è vuoto (linea 24110) e visualizzare l'eventuale messaggio di avvertimento (linee 24130÷24150). Inizia poi un ciclo (linee 24180÷24460) in cui, dopo essersi posizionato sul record da stampare, il programma visualizza il contenuto dei campi cognome, nome e numero di telefono e, in seguito alla digitazione del tasto RETURN, legge e stampa il contenuto del record successivo, dopo aver cancellato le linee video utilizzate. L'uscita dal ciclo avviene quando si incontra il primo record libero, quindi è visualizzato un messaggio (linee 24490÷24510) e si chiude il canale utilizzato (linea 24530).
La fine della sessione di lavoro è gestita dalla routine 25000 che chiude il canale di comando prima di terminare l'esecuzione del programma.

```
60080 REM ACCENSIONE E SPEGNIMENTO DEL CURSORE
60100 IF Z7<TI AND NOT(Z6) THEN PRINT "▒▌";:Z6=NOT(Z6):Z7=TI+15
60110 IF Z7<TI AND Z6 THEN PRINT " ▌";:Z6=NOT(Z6):Z7=TI+15
60120 GOTO 60060
60140 REM E' STATO DIGITATO UN CARATTERE
60160 Z8=ASC(Z8$):Z9=LEN(IN$)
60180 REM SE NON E' UN CARATTERE ALFANUMERICO, DEVE ESSERE UN RETURN O DELETE
60200 IF NOT(Z8>31 AND Z8<96) THEN GOTO 60320
60220 REM CONTROLLO CHE NON SIA STATA SUPERATA LA LUNGHEZZA MASSIMA
60240 IF Z9=ZL THEN GOTO 60060
60260 REM LO AGGIUNGO ALLA STRINGA IN$
60280 IN$=IN$+Z8$:PRINT Z8$;:GOTO 60060
60300 REM SE E' UN RETURN, HO TERMINATO LA LETTURA
60320 IF Z8=13 THEN PRINT " ▌";:RETURN
60340 REM SE E' DELETE, CANCELLO IN IN$ E SUL VIDEO L' ULTIMO CARATTERE DIGITATO
60360 IF Z8=20 AND Z9>0 THEN IN$=LEFT$(IN$,Z9-1):PRINT " ▌▌";:GOTO 60060
60370 GOTO 60060
60960 REM ROUTINE:INIZIALIZZAZIONE COSTANTI
60980 REM CIRCUITO VIDEO
61000 VI=53248
61020 REM CIRCUITO SUONO
61040 SI=54272
61060 REM MEMORIA VIDEO
61080 MV=1024
61100 REM MEMORIA COLORE
61120 MC=55296
61140 REM COSTANTI DI USO COMUNE
61160 ZL=9
61180 REM INIZIALIZZAZIONE CHIP SUONO
61200 FOR I=0 TO 24
61210 POKE SI+I,0
61220 NEXT I
61230 RETURN
61980 REM ROUTINE:STAMPA IL TITOLO DEL PROGRAMMA MEMORIZZATO IN PG$
62000 GOSUB 61000
62010 POKE VI+32,15
62020 POKE VI+33,15
62030 PRINT "▒▒▒▒▒";
62040 PRINT TAB(6);" ▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬ "
62050 FOR I=1 TO 5
62060 PRINT TAB(6);"▒▒                         ▌"
62070 NEXT I
62080 PRINT TAB(6);" ▒▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▬▒ "
62090 PRINT TAB(6);"▒▒▒▒▒▒▒DIGITA ▒RETURN▒ PER PROSEGUIRE"
62100 PRINT "▒▒▒▒▒"
62110 FOR I=1 TO 5
62120 PRINT TAB(7);
62130 FOR J=1 TO 26
62140 PRINT "▒▒ ";
62150 NEXT J
62160 PRINT
62170 NEXT I
62190 REM ORA SCRIVO IL TITOLO
62210 PRINT "▒▒▒▒▒▒▒▒▒▒";TAB((40-LEN(PG$))/2);"▒▒";PG$
62220 GET Z9$
62230 IF Z9$<>CHR$(13) THEN GOTO 62220
62240 PRINT "▒▒";
62250 RETURN
```

Aritmetica a precisione infinita

I computer, per quanto abbiano enormi possibilità, hanno pur sempre i loro limiti. Nel CBM-64, ad esempio, i numeri hanno nove cifre di precisione; il Basic arrotonda le ulteriori cifre significative per difetto se la decima cifra è 4 o meno di 4, per eccesso se è 5 o più di 5.
Volendo effettuare l'operazione 999·999·999 + 999·999·996, il CBM-64 darà come risultato 2 E+9, cioè rappresenterà il numero in notazione esponenziale. Traducendo la notazione esponenziale in notazione decimale, avremo

$$2\ E+9=2*10^9=2*1\cdot000\cdot000\cdot000=2\cdot000\cdot000\cdot000.$$

Il risultato esatto della somma è però 1·999·999·995. Anche usando la notazione esponenziale, c'è sempre un limite alla grandezza dei numeri che il Basic del CBM-64 è in grado di trattare. Progetteremo quindi un programma per effettuare operazioni senza che i numeri siano arrotondati, cioè senza che i risultati delle operazioni siano, di conseguenza, falsati.

Analisi del problema

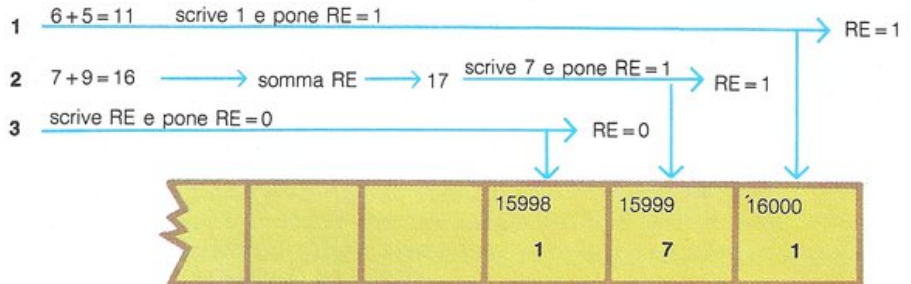
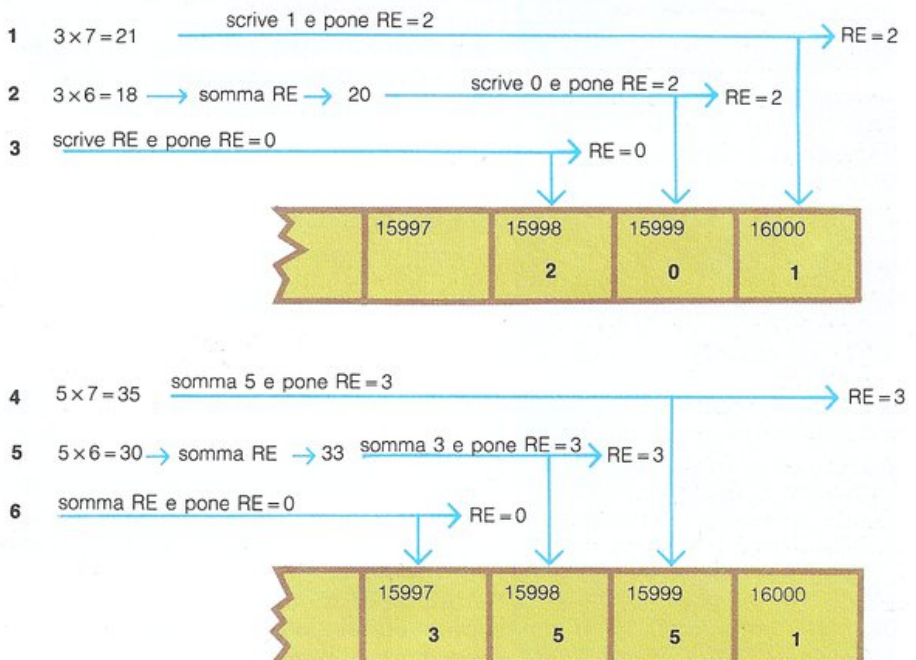
Come sappiamo, il CBM-64 assegna 4 bytes di memoria alle variabili e alle costanti numeriche reali (in virgola mobile), e 2 bytes a quelle intere. Ogni byte è composto da 8 bits, e quindi, poiché i calcolatori utilizzano il sistema binario, il numero intero più grande memorizzabile in 2 bytes è $2^{16}-1=65\cdot535$. In realtà le costanti e le variabili intere possono assumere valori compresi nel campo $-32\cdot768 \div +32\cdot767$, perché il bit più significativo è riservato al segno (positivo o negativo).
Per superare questi limiti è necessario utilizzare un diverso metodo di memorizzazione dei numeri.
Il nostro programma dovrà permettere di eseguire l'addizione e la moltiplicazione di due numeri interi di qualsiasi dimensione senza arrotondare il risultato.
Come prima cosa sarà presentata la possibilità di scegliere tra l'addizione (+), la moltiplicazione (*) o l'interruzione del programma (F).
Quindi, il primo numero, che chiameremo primo operando, sarà memorizzato in una stringa, e poi ogni singola cifra che compone il numero sarà memorizzata in una diversa cella di memoria, a partire dalla locazione di indirizzo 2048. Il secondo numero (secondo operando) sarà poi memorizzato nelle celle immediatamente successive a quella contenente l'ultima cifra del primo. Per individuare la prima cella di memoria del secondo operando, basterà aggiungere 2048 al numero delle cifre del primo. L'area riservata alla memorizzazione dei due numeri e del risultato sarà compresa tra l'indirizzo 2048 e l'indirizzo 16000, permettendo così, in teoria, di effettuare operazioni tra numeri interi composti di molte migliaia di cifre.

Per effettuare la somma di due numeri, il programma dovrà riportare i contenuti delle due celle di memoria in cui sono memorizzate le cifre meno significative (le unità) dei due numeri in altrettante variabili di comodo. Le dovrà poi sommare tra di loro, controllare se il risultato è maggiore di 9, e se lo è mettere la cifra 1 nella variabile RE (riporto), prima di memorizzare con una POKE il risultato della somma (a cui si sarà sottratto 10) nell'indirizzo 16000 (TP). Se la somma non è maggiore di 9, il risultato sarà memorizzato in TP senza dover considerare il riporto.
Saranno poi sommati tra di loro i contenuti delle celle di memoria che rappresentano le decine, aggiungendovi l'eventuale riporto della somma precedente, e il risultato sarà memorizzato in TP – 1. Si procederà in que-

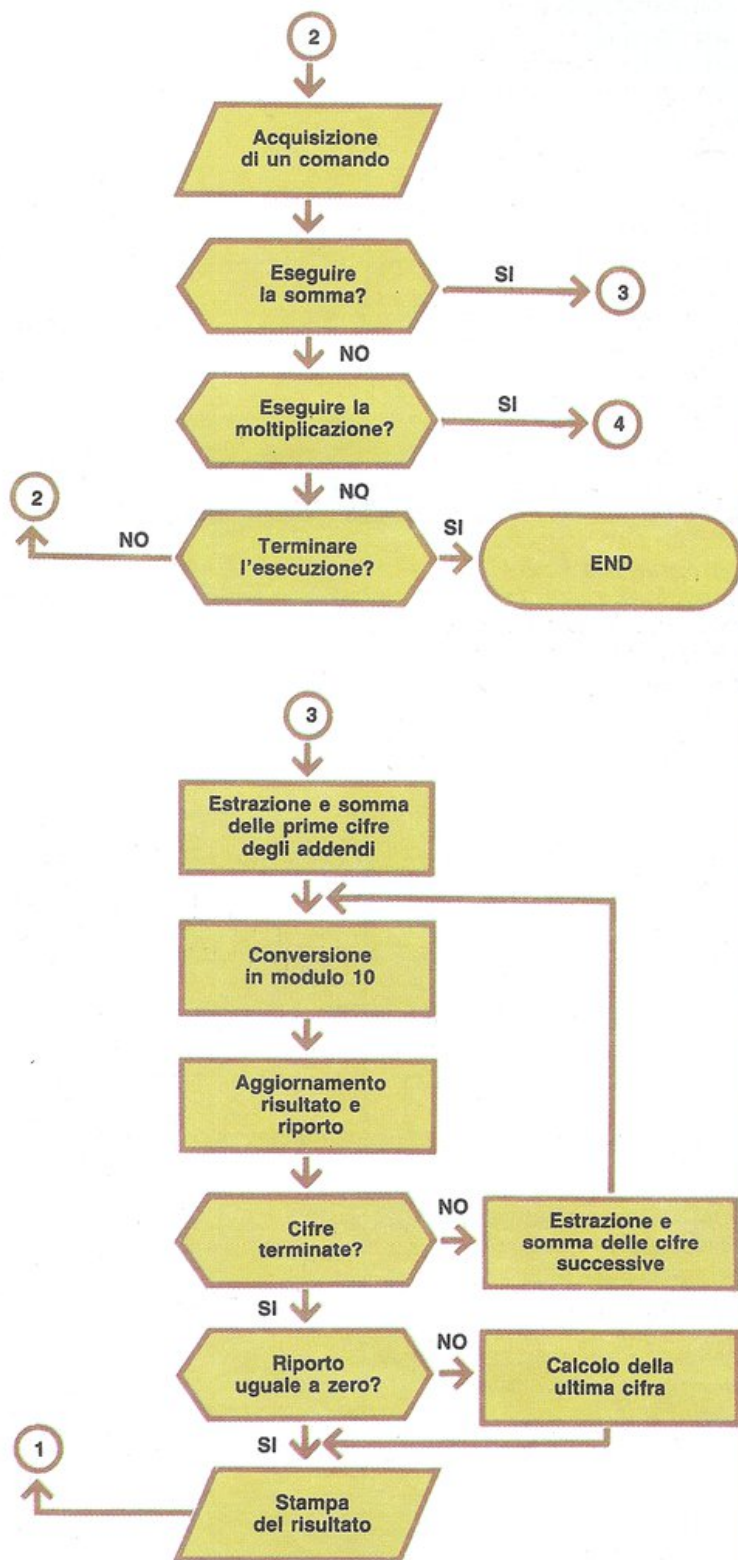
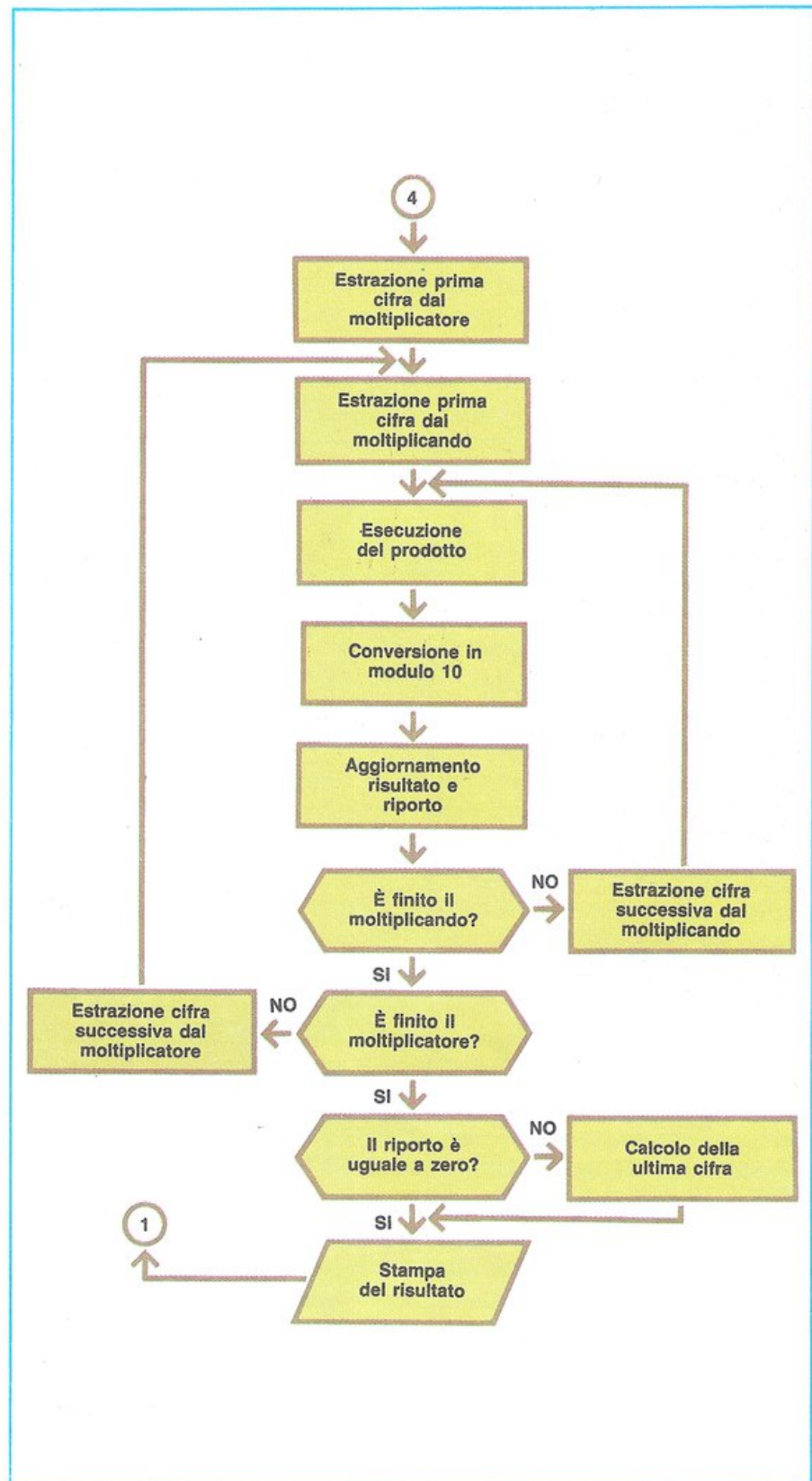
```
ARITMETICA A PRECISIONE INFINITA

Hai a disposizione due operazioni,+ e *
che pero' puoi eseguire tra numeri
interi di lunghezza qualsiasi.
Per eseguire i calcoli occorre
dare i valori dei due operandi ed il
tipo dell'operazione che si vuole
eseguire. Se non si da' un operando,
questo viene sostituito dal risultato
della precedente operazione.

RETURN per proseguire.
```

```
ARITMETICA A INFINITA PRECISIONE

operando 1 ? 2345637859765342764975

operando 2 ? 1524376648579435778398
```

```
ARITMETICA A INFINITA PRECISIONE

operando 1 ?

operando 2 ? 1524376648579435778398

Operazione [+,*,F] ? Somma

Risultato:
3870014508344778543373
```

sto modo fino a che non sarà stata calcolata l'intera somma. Il programma dovrà inoltre ricordare di controllare anche l'esistenza di un eventuale ultimo riporto.
Questo procedimento, illustrato qui di seguito con un esempio, è in pratica quello a cui siamo tutti abituati.

Per effettuare la moltiplicazione, il programma dovrà invece moltiplicare la cifra meno significativa del primo numero per tutte quelle del secondo, tenendo conto dei riporti, e memorizzare i relativi risultati dall'indirizzo 16000 al 2048. La successiva cifra dell'operando sarà moltiplicata per quelle dell'operatore, ed i risultati andranno memorizzati negli indirizzi dal 15999 al 2048. Nella figura qui sotto è mostrato un esempio.

Diagrammi di flusso

Il programma inizia controllando che la sua base in memoria sia stata spostata. Tale necessità scaturisce dall'esigenza di riservare maggiore spazio alla memorizzazione degli operandi.
Successivamente, dopo aver inizializzato le costanti e stampato il titolo, si visualizzano i campi in cui saranno immessi gli operandi, quello in cui apparirà il risultato e quello in cui sono elencate le operazioni consentite.
Quindi il programma attende che l'utente attribuisca un valore al primo operando; la semplice pressione del tasto RETURN farà sì che il programma attribuisca al primo operando il risultato della precedente operazione.
L'eventuale operando immesso è memorizzato in un gruppo di locazioni di memoria adiacenti.
Le operazioni descritte per il ciclo di determinazione del primo operando sono ripetute per la determinazione del secondo.

Una volta stabilito il valore degli operandi, il programma attende istruzioni dall'utente per attivare la somma o la moltiplicazione, o per terminare l'esecuzione. Il comando immesso sarà utilizzato all'interno di un ciclo per eseguire l'operazione desiderata dall'utente.
Se l'operazione da eseguire è una somma, si estrae l'ultima cifra (a destra) di entrambi gli operandi, poi si esegue la somma delle cifre estratte e si converte il risultato in modulo 10 (cioè si divide per 10 e si considera il resto di tale divisione, che sarà compreso fra 0 e 9).
L'eventuale riporto è memorizzato per essere sommato al risultato dell'addizione delle cifre successive.
Aggiornati la somma e il riporto, si verifica se è stata effettuata la somma di tutte le cifre degli addendi; se non è così si passa a considerare le cifre successive, altrimenti si controlla il valore del riporto ed eventualmente lo si aggiunge, come ulteriore cifra, al risultato.

Nel caso sia richiesta la moltiplicazione si estrae una cifra dal moltiplicando e una dal moltiplicatore, se ne esegue il prodotto, che sarà convertito in modulo 10, per aggiornare il risultato e il riporto. Si esegue quindi una doppia verifica atta a determinare se sono finite le cifre dei due operandi; eventualmente non siano finite si torna a considerare le rimanenti. Una volta moltiplicate tutte le cifre del moltiplicando e del moltiplicatore, e aggiornato il risultato, si verifica il valore del riporto e si stampa il risultato. All'ultimo blocco del ciclo è demandato il controllo della possibilità di uscita dall'esecuzione del programma.

Il programma

Il programma inizia alla linea 81, in cui controlla se è stata spostata la sua base di memoria. Segue l'inizializzazione delle costanti e dei colori dello sfondo e della cornice (linee 83÷85). Sono quindi inizializzate alcune ulteriori costanti (linea 160). In particolare si stabilisce il valore delle costanti PI e TP che individuano il primo e l'ultimo indirizzo di memoria dell'area che sarà utilizzata per la memorizzazione degli operandi e del risultato. Le linee 200÷310 presentano la spiegazione delle possibilità offerte dal programma. La linea 342 passa poi il controllo alla routine 3000 che, visualizzata la richiesta di inserire il primo operando e, memorizzato il valore di PI in OP e in O1, chiama la 5000. Il programma quindi controlla se è stato effettivamente immesso un numero (linea 5010) e, tolti gli eventuali zeri non significativi (linea 5121), lo memorizza nelle celle di memoria dall'indirizzo 2048 in poi (linee 5130÷5150). Qualora invece non sia stato digitato un numero, ma il tasto RETURN, il programma copia il risultato dell'operazione precedente nell'area di memoria destinata all'operando attuale (linee 5030÷5050), e lo visualizza sullo schermo (linee 5062÷5065). Restituisce poi il controllo alla linea 3080, dove memorizza il numero di cifre del primo operando in L1, e torna alla linea 541, che permette di eseguire lo

```
0 REM ********************
1 REM *PRECISIONE INFINITA*
2 REM ********************
4 REM VARIABILI UTILIZZATE
6 REM I,J        :CONTATORI DI CICLO
7 REM ZE         :COSTANTE=0
8 REM TP         :TOP DELLA MEMORIA DISPONIBILE(=16000) E INDIRIZZO DEL RISULTATO
9 REM PI         :BASE DELLA MEMORIA DISPONIBILE=2048
10 REM R$        :CONTIENE I CARATTERI REVERSE-ON E REVERSE-OFF
11 REM C         :COSTANTE=1
12 REM IN$       :CONTIENE LE STRINGHE DIGITATE
13 REM Z9        :LUNGHEZZA DELLA STRINGA IN$
14 REM OP        :INDIRIZZO DI PARTENZA DELL'OPERANDO IN ESAME
15 REM O1,O2     :INDIRIZZI DI PARTENZA DEI DUE OPERANDI
16 REM LU        :NUMERO DI CIFRE DELL'OPERANDO IN ESAME
17 REM L1,L2     :NUMERO DI CIFRE DEI DUE OPERANDI
18 REM T         :USATA NEL CALCOLO DEL TEMPO DURANTE LA FASE DI LAMPEGGIO
19 REM OP$       :NOME DELL'OPERAZIONE RICHIESTA
20 REM MA        :MASSIMO TRA L1 E L1
21 REM LR        :NUMERO DI CIFRE DEL RISULTATO
22 REM A1,A2,BU  :BUFFER DI COMODO
80 REM CONTROLLO CHE IL PROGRAMMA SIA STATO CARICATO ALLA GIUSTA LOCAZIONE
81 IF PEEK(44)=64 THEN GOTO 101
82 REM AVVERTO CHE IL PROGRAMMA NON PUO' ANDARE IN ESECUZIONE
83 GOSUB 61000:REM INIZIALIZZO COSTANTI C64
84 POKE VI+32,6:REM CORNICE CELESTE
85 POKE VI+33,6:REM SFONDO CELESTE
86 PRINT "▒":REM CARATTERI IN GRIGIO
87 PRINT "▒ATTENZIONE, HAI DIMENTICATO"
88 PRINT "▒▒DI IMMETTERE I SEGUENTI COMANDI:"
89 PRINT "▒▒▒1) POKE 44,64"
90 PRINT "▒▒2) POKE 16384,0"
91 PRINT "▒▒PRIMA DI IMMETTERLI RICORDA"
92 PRINT "▒▒DI SALVARE IL PROGRAMMA, SE QUESTO "
93 PRINT "▒NON E' GIA' PRESENTE SU NASTRO,"
94 PRINT "▒E POI DI RICARICARLO"
96 PRINT "▒▒"
97 END
100 REM TITOLO ED INIZIALIZZAZIONE COSTANTI C64
101 PRINT CHR$(142):REM CARATTERI MAIUSCOLI
110 PG$="PRECISIONE INFINITA"
120 GOSUB 62000
130 PRINT "■":REM CARATTERI NERI
140 PRINT CHR$(14):REM CARATTERI MINUSCOLI
150 REM INIZIALIZZO COSTANTI DI USO
160 ZE=0:PI=2048:TP=16000:R$="▒▒":C=1
190 REM SPIEGAZIONE PROGRAMMA
200 PRINT "▒";
210 PRINT "     ♠—╮|╲‾|╮‾♠ ♠ ┐_‾‾╮♥╮┌╱‾ ╮╱—╮╱╲|♠"
220 PRINT "▒HAI A DISPOSIZIONE DUE OPERAZIONI,+ E *"
230 PRINT "▒CHE PERO' PUOI ESEGUIRE TRA NUMERI"
240 PRINT "▒INTERI DI LUNGHEZZA QUALSIASI."
250 PRINT "▒▒ER ESEGUIRE I CALCOLI OCCORRE"
260 PRINT "▒DARE I VALORI DEI DUE OPERANDI ED IL"
270 PRINT "▒TIPO DELL'OPERAZIONE CHE SI VUOLE"
280 PRINT "▒ESEGUIRE. ♥E NON SI DA' UN OPERANDO,"
290 PRINT "▒QUESTO VIENE SOSTITUITO DAL RISULTATO"
300 PRINT "▒DELLA PRECEDENTE OPERAZIONE."
305 PRINT"▒▒ER FINIRE DEVI DIGITARE ▒_▒."
310 PRINT "▒▒_‾| ╱—╱▒ PER PROSEGUIRE."
320 GET IN$:IF IN$<>CHR$(13) THEN GOTO 320
330 REM INIZIA IL PROGRAMMA
340 PRINT "▒    ♠—╮|╲‾|╮‾♠ ♠ ╮╱—╮╱╲|♠ ┐_‾‾╮♥╮┌╱‾"
341 REM LEGGO IL PRIMO OPERANDO
342 GOSUB 3000
540 REM LEGGO IL SECONDO OPERANDO
541 GOSUB 3500
670 REM LEGGO IL CODICE DELL'OPERAZIONE DA ESEGUIRE
680 REM FACCIO LAMPEGGIARE LA SCRITTA FINCHE' NON E' DATO UN INPUT CORRETTO
690 REM LA SCRITTA RIMANE IN UNO STATO NORMALE O DI REVERSE PER 1/2 DI SECONDO
720 T=TI
754 POKE 214,13:PRINT:REM RIGA 14
760 PRINT MID$(R$,C,1);"┌PERAZIONE [+,*,_] ?▒                    ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒";
770 GET IN$:IF IN$="" THEN GOTO 850
780 REM VEDO SE IL COMANDO E' CORRETTO, SE LO E' STAMPO IL CODICE OPERAZIONE
790 IF IN$="+" THEN OP$="♥OMMA":GOSUB 4000:GOSUB 10000:GOTO 341
800 IF IN$="*" THEN OP$="\OLTIPLICAZIONE":GOSUB 4000:GOSUB 11000:GOTO 341
801 IF IN$="F" THEN PRINT "▒▒▒";CHR$(142)
802 IF IN$="F" THEN POKE53280,14:POKE53281,6:PRINT"▒▒":END
830 REM IL COMANDO E' ERRATO
840 GOTO 754
850 IF TI>T+30 THEN C=3-C:T=TI
860 GOTO 754
2990 REM ROUTINE:LEGGE IL VALORE DEL PRIMO OPERANDO
3000 POKE 214,5:PRINT:REM MI POSIZIONO SULLA SESTA RIGA
3010 PRINT  "OPERANDO 1 ? ";
3020 REM LEGGO OPERANDO
3021 OP=PI:O1=OP
3030 GOSUB 5000
3080 L1=LU:REM AGGIORNO IL NUMERO DI CIFRE DEL PRIMO OPERANDO
```

*stesso procedimento per il secondo operando. Questo è memorizzato a partire dalla cella immediatamente successiva a quella occupata dall'ultima cifra del primo operando, individuata aggiungendo all'indirizzo 2048 il numero delle cifre che compongono il primo operando (linea 3521). È poi presentata la richiesta di scegliere un'operazione (linea 760). Qualora si digiti il tasto F il programma termina (linea 801); se invece si premono i tasti + o * (linee 790÷800), sono visualizzate le scritte «somma» o «moltiplicazione» e «risultato» (routine 4000), ed è cancellata la zona del video dove il risultato sarà stampato (routine 21000). A seconda dell'operazione scelta, si passa il controllo alla routine 10000 (che effettua la somma) o alla 11000 (che esegue la moltiplicazione). Per effettuare la somma, è innanzitutto necessario memorizzare in MA (variabile utilizzata per determinare quante volte sarà eseguito il ciclo della somma) il numero delle cifre che compongono l'addendo più lungo (linee 10000÷10010). Sono poi inizializzate le variabili LR (numero di cifre del risultato) e RE (riporto), ed è eseguito il ciclo della somma (linee 10040÷10080). Questo inizia incrementando la lunghezza del risultato (linea 10050) e riportando in due variabili di comodo (A1 e A2) il contenuto degli indirizzi dove erano state memorizzate le unità dei due numeri da sommare. Vengono poi sommati A1, A2 e il riporto (che nel caso*

```
3090 RETURN
3500 POKE 214,9:PRINT:REM MI POSIZIONO SULLA NONA RIGA
3510 PRINT  "OPERANDO 2 ? ";
3520 REM LEGGO OPERANDO
3521 OP=PI+L1:O2=OP
3530 GOSUB 5000
3580 L2=LU:REM AGGIORNO IL NUMERO DI CIFRE DEL SECONDO OPERANDO
3590 RETURN
3990 REM ROUTINE:STAMPO IL CODICE DELL'OPERAZIONE DA ESEGUIRE
4000 POKE 214,13:PRINT
4010 PRINT "■PERAZIONE [+,*,_] ?                    ■■■■■■■■■■■■■■■■■■";
4060 PRINT OP$
4070 PRINT "■■_ISULTATO:"
4071 PRINT "‾‾‾‾‾‾‾‾‾‾‾"
4072 POKE 214,17:PRINT:REM RIGA 18
4073 GOSUB 21000:REM CANCELLO PARTE DELLE RIGHE PER IL RISULTATO
4080 RETURN
4990 REM ROUTINE:LEGGE UN OPERANDO,SE E'="" LO PONE PARI AL RISULTATO PRECEDENTE
5000 ZL=100:GOSUB 60000
5010 IF Z9>0 THEN GOTO 5121:REM LEN(IN$)=0?
5020 REM COPIO RISULTATO PRECEDENTE IN OPERANDO ATTUALE
5030 FOR I=0 TO LR-1
5040 POKE OP+I,PEEK(TP-LR+I+1)
5050 NEXT I
5060 LU=LR:REM AGGIORNO LUNGHEZZA
5061 REM ORA STAMPO IL VALORE MEMORIZZATO
5062 FOR I=OP TO OP+LU-1
5063 PRINT RIGHT$(STR$(PEEK(I)),1);
5064 NEXT I
5065 PRINT
5070 RETURN
5080 REM L'OPERANDO E' STATO DIGITATO
5120 REM TOLGO EVENTUALI ZERI NON SIGNIFICATIVI
5121 IF Z9>1 THEN IF LEFT$(IN$,1)="0" THEN Z9=Z9-1:IN$=RIGHT$(IN$,Z9):GOTO 5121
5122 REM COPIO IN$ IN MEMORIA AL POSTO DELL'OPERANDO OP
5130 FOR I=1 TO Z9
5140 POKE OP+I-1,VAL(MID$(IN$,I,1))
5150 NEXT I
5160 LU=Z9
5200 RETURN
9990 REM ROUTINE:ESEGUE L'OPERAZIONE DI SOMMA TRA I DUE OPERANDI
10000 MA=L1:REM CALCOLO IL MASSIMO TRA LE DUE LUNGHEZZE
10010 IF MA<L2 THEN MA=L2
10021 REM ESEGUO LA SOMMA
10030 LR=0:RE=0
10040 FOR I=1 TO MA
10050 LR=LR+1
10051 A1=PEEK(O1+L1-I):IF I>L1 THEN A1=0
10052 A2=PEEK(O2+L2-I):IF I>L2 THEN A2=0
10060 POKE TP-I+1,A1+A2+RE
10070 RE=0:IF PEEK(TP-I+1)>9 THEN POKE(TP-I+1),PEEK(TP-I+1)-10:RE=1
10080 NEXT I
10090 REM CONTROLLO IL RESTO
10100 IF RE=0 THEN GOTO 10150
10110 LR=LR+1
10140 POKE TP-I+1,1
10141 REM STAMPO IL RISULTATO
10150 GOSUB 20000
10160 RETURN
10990 REM ROUTINE:ESEGUE L'OPERAZIONE DI MOLTIPLICAZIONE TRA I DUE OPERANDI
11000 FOR I=TP TO TP-L1-L2+1 STEP -1
11001 POKE I,ZE
11002 NEXT I
11003 REM HO AZZERATO LA ZONA DI MEMORIA IN CUI ANDRA' IL RISULTATO
11010 RE=0:LR=0
11020 FOR I=0 TO L1-1
11030 FOR J=0 TO L2-1
11031 BU=TP-I-J
11032 IF LR-1<I+J THEN LR=LR+1
11033 IF PEEK(O1+L1-I-1)=0 THEN J=L2:GOTO 11060
11040 POKE BU,PEEK(O1+L1-I-1)*PEEK(O2+L2-J-1)+RE+PEEK(BU)
11051 RE=INT(PEEK(BU)/10)
11052 POKE BU,PEEK(BU)-RE*10
11060 NEXT J
11061 IF RE=0 THEN GOTO 11070
11062 POKE BU-1,RE-INT(RE/10)*10
11063 RE=INT(RE/10)
11064 BU=BU-1:IF LR-1<TP-BU THEN LR=LR+1
11065 GOTO 11061
11070 NEXT I
11080 REM CONTROLLO UN RESTO SULL'ULTIMA CIFRA
11090 IF RE=0 THEN GOTO 11150
11100 POKE TP-LR,RE-INT(RE/10)*10
11110 RE=INT(RE/10)
11120 LR=LR+1
11130 GOTO 11090
11140 REM STAMPO IL RISULTATO
11150 GOSUB 20000
11160 RETURN
19990 REM ROUTINE:STAMPA IL RISULTATO CONTENUTO IN MEMORIA
```

delle unità sarà 0), mettendo il risultato nell'indirizzo di memoria TP (linea 10060). Qualora il risultato di tale somma sia maggiore di 9, si memorizza 1 in RE e al contenuto della memoria di indirizzo TP (risultato) è sottratto 10 (linea 10070). Il ciclo è poi ripetuto per le decine, memorizzando il risultato nella locazione TP – 1, e così di seguito, fino a quando non si sono sommate tutte le cifre che componevano i due addendi. Si controlla poi se è rimasto un riporto (linee 10100÷10140), prima di visualizzare il risultato ricorrendo alla routine 20000.
La routine 11000, che esegue la moltiplicazione di due numeri, inizia con l'azzeramento di tutte le celle in cui andrà memorizzato il risultato dell'operazione (linee 11000÷11002), passando poi all'inizializzazione delle variabili RE e LR (linea 11010). Si incontra quindi il doppio ciclo che esegue la moltiplicazione (linee 11020÷11070), nel quale si moltiplica innanzitutto la cifra meno significativa del primo operando per tutte le cifre del secondo (linee 11031÷11060), tenendo sempre conto dei riporti (linee 11061÷11070), poi la successiva cifra del primo operando per tutte quelle del secondo, e così via fino ad esaurire le cifre da moltiplicare. Ogni risultato è memorizzato in un indirizzo di memoria ed i risultati che corrispondono ad uno stesso indirizzo sono sommati tra di loro. Si controlla quindi l'eventuale riporto (linee 11090÷11130), prima di visualizzare il risultato.

```
20000 IF PEEK(TP-LR+1)=0 THEN IF LR>1 THEN LR=LR-1:GOTO 20000
20020 REM HO TOLTO GLI EVENTUALI ZERI NON SIGNIFICATIVI
20100 POKE 214,18:PRINT
20110 FOR I=TP-LR+1 TO TP
20120 PRINT RIGHT$(STR$(PEEK(I)),1);
20130 NEXT I
20140 PRINT
20150 RETURN
20990 REM ROUTINE:CANCELLA PARTE DELLE RIGHE NELLE QUALI APPARIRA' IL RISULTATO
21000 FOR I=MV+19*40 TO MV+25*40-1
21010 POKE I,32
21020 NEXT I
21030 RETURN
59920 REM ROUTINE:GESTISCE L'INPUT DI UNA STRINGA NUMERICA
59930 REM LE VARIABILI UTILIZZATE SONO:Z6,Z7,Z8,Z9,Z8$,IN$
59940 REM IN INGRESSO VUOLE IL VALORE ZL CHE E' LA LUNGHEZZA MASSIMA DELLA
59950 REM STRINGA DA LEGGERE
59960 REM IN USCITA DA' LA STRINGA LETTA IN IN$ E LA SUA LUNGHEZZA IN Z9
59980 REM CANCELLO LA ZONA DI SCHERMO IN CUI VERRA' DIGITATA LA STRINGA
60000 FOR Z8=1 TO ZL: PRINT " ";:NEXT Z8
60010 FOR Z8=1 TO ZL: PRINT "▮";:NEXT Z8
60020 IN$="":Z7=TI
60040 REM LEGGO UN CARATTERE
60060 GET Z8$:IF Z8$<>"" THEN 60160
60080 REM ACCENSIONE E SPEGNIMENTO DEL CURSORE
60100 IF Z7<TI AND NOT(Z6) THEN PRINT "▒▮";:Z6=NOT(Z6):Z7=TI+15
60110 IF Z7<TI AND Z6 THEN PRINT " ▮";:Z6=NOT(Z6):Z7=TI+15
60120 GOTO 60060
60140 REM E' STATO DIGITATO UN CARATTERE
60160 Z8=ASC(Z8$):Z9=LEN(IN$)
60180 REM SE NON E' UN CARATTERE NUMERICO, DEVE ESSERE UN RETURN O DELETE
60190 IF NOT(Z8>47 AND Z8<58) THEN GOTO 60320
60200 IF NOT((Z8>47 AND Z8<58) OR (Z8>64 AND Z8<91)) THEN GOTO 60320
60220 REM CONTROLLO CHE NON SIA STATA SUPERATA LA LUNGHEZZA MASSIMA
60240 IF Z9=ZL THEN GOTO 60060
60260 REM LO AGGIUNGO ALLA STRINGA IN$
60280 IN$=IN$+Z8$:PRINT Z8$;:GOTO 60060
60300 REM SE E' UN RETURN, HO TERMINATO LA LETTURA
60320 IF Z8=13 THEN PRINT " ▮";:RETURN
60340 REM SE E' DELETE, CANCELLO IN IN$ E SUL VIDEO L' ULTIMO CARATTERE DIGITATO
60360 IF Z8=20 AND Z9>0 THEN IN$=LEFT$(IN$,Z9-1):PRINT " ▮▮";:GOTO 60060
60370 GOTO 60060
60960 REM ROUTINE:INIZIALIZZAZIONE COSTANTI
60980 REM CIRCUITO VIDEO
61000 VI=53248
61020 REM CIRCUITO SUONO
61040 SI=54272
61060 REM MEMORIA VIDEO
61080 MV=1024
61100 REM MEMORIA COLORE
61120 MC=55296
61140 REM COSTANTI DI USO COMUNE
61160 ZL=9
61180 REM INIZIALIZZAZIONE CHIP SUONO
61200 FOR I=0 TO 24
61210 POKE SI+I,0
61220 NEXT I
61230 RETURN
61980 REM ROUTINE:STAMPA IL TITOLO DEL PROGRAMMA MEMORIZZATO IN PG$
62000 GOSUB 61000
62010 POKE VI+32,15
62020 POKE VI+33,15
62030 PRINT "▒▒▒▒▒";
62040 PRINT TAB(6);" ┌──────────────────────────┐"
62050 FOR I=1 TO 5
62060 PRINT TAB(6);"│                          │"
62070 NEXT I
62080 PRINT TAB(6);" └──────────────────────────┘"
62090 PRINT TAB(6);"▒▒▒▒▒▒▒DIGITA ▒RETURN▒ PER PROSEGUIRE"
62100 PRINT "▒▒▒▒▒"
62110 FOR I=1 TO 5
62120 PRINT TAB(7);
62130 FOR J=1 TO 26
62140 PRINT "▒▒ ";
62150 NEXT J
62160 PRINT
62170 NEXT I
62190 REM ORA SCRIVO IL TITOLO
62210 PRINT "▒▒▒▒▒▒▒▒▒";TAB((40-LEN(PG$))/2);"▒▒";PG$
62220 GET Z9$
62230 IF Z9$<>CHR$(13) THEN GOTO 62220
62240 PRINT "▒▒";
62250 RETURN
```

L'archivio veloce

Gli elenchi di dati costituiscono probabilmente le strutture informative più impiegate nelle applicazioni di ogni tipo. Tali elenchi possono essere caratterizzati in modo preciso dando loro il nome di tabelle. Una tabella è una successione di elementi, ciascuno dei quali comprende due parti: una **chiave** che distingue l'elemento dagli altri e un'**informazione** associata alla chiave. Funzione della tabella è quindi la memorizzazione di dati cui successivamente si può accedere mediante le chiavi associate.
L'operazione fondamentale che si esegue di solito in una tabella è la ricerca di un elemento, ma bisogna anche tener conto della eventuale necessità di effettuare inserzioni e cancellazioni, e delle modalità di individuazione degli elementi.
I principali metodi utilizzati sono la ricerca completa (sequenziale), la ricerca binaria e la ricerca **hash**.
Il programma che vogliamo realizzare sarà un esempio di gestione di elenchi di dati con metodo di ricerca hash.

Analisi del problema

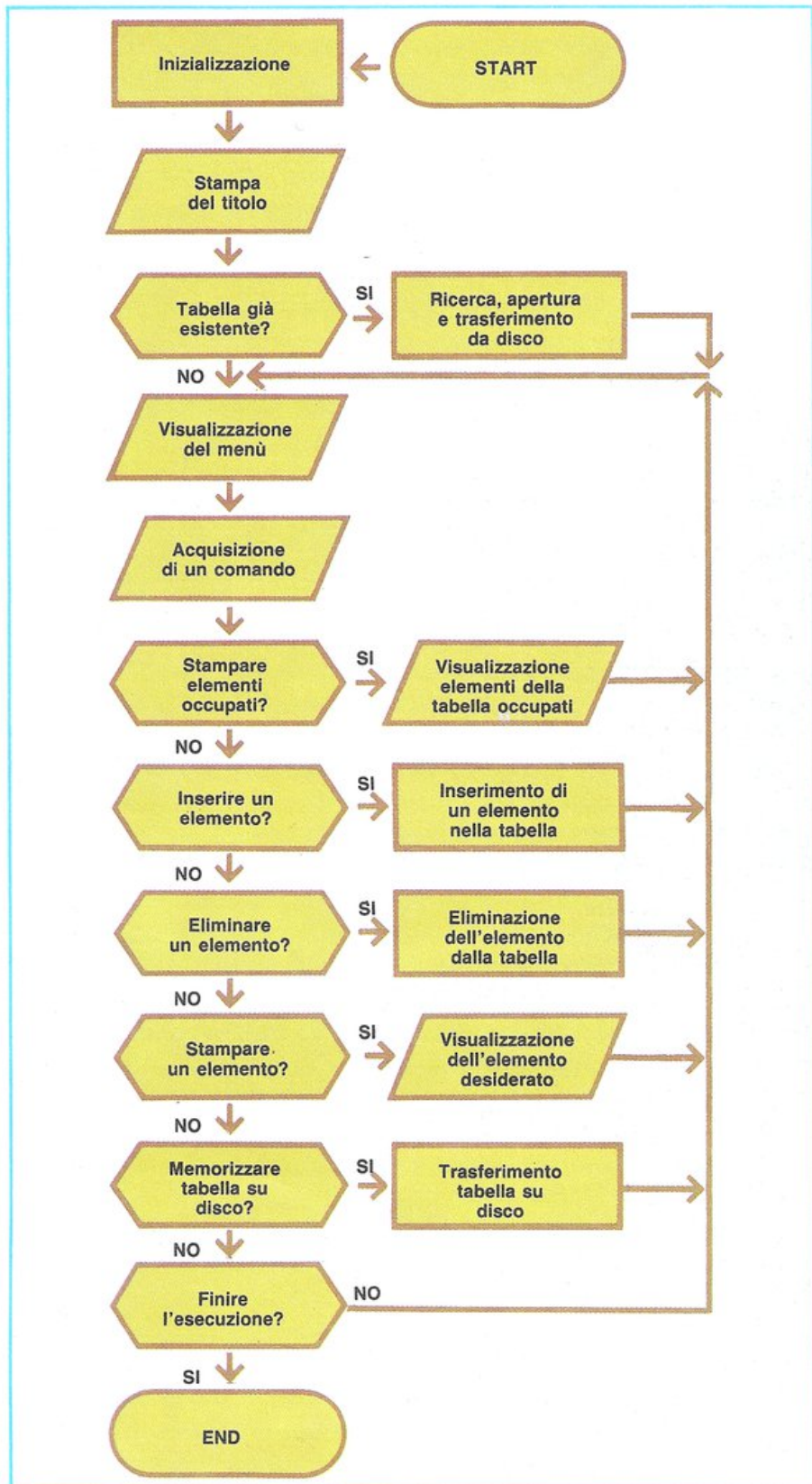
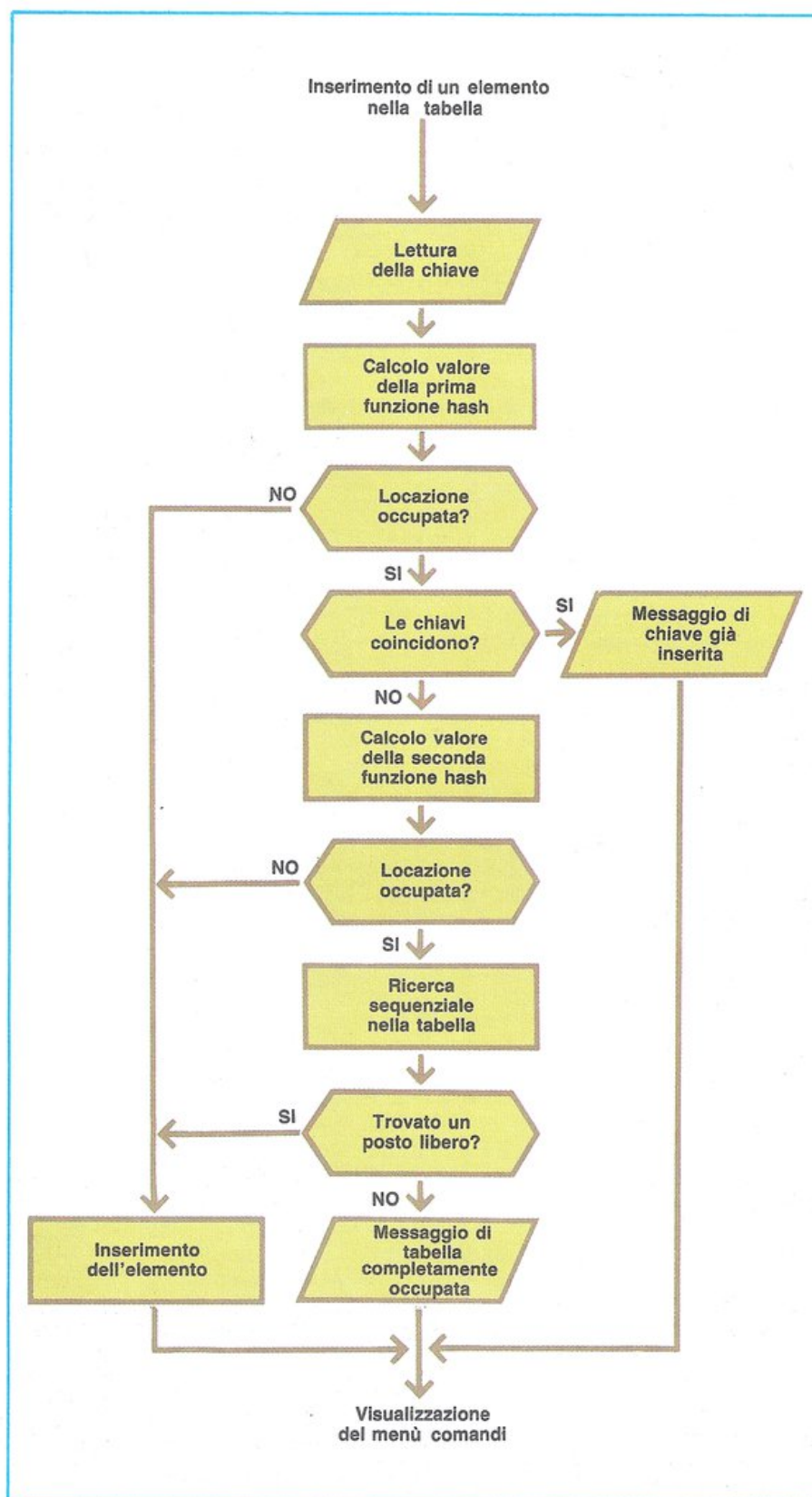
Il metodo più semplice per organizzare una tabella è quello di memorizzare i suoi elementi senza seguire alcuna regola di ordinamento. L'operazione di ricerca nella tabella, detta ricerca completa, si effettua scandendone gli elementi fino a trovare la chiave desiderata. Il numero medio di confronti necessari a individuare un elemento con questo sistema è (n+1)/2, con n uguale al numero degli elementi della tabella. Ovviamente quanto più grande è la tabella tanto maggiore sarà il numero dei confronti necessari per individuare un elemento; ciò significa che le tabelle organizzate con questo metodo sono utili solo per memorizzare una quantità limitata di elementi.
La ricerca in una tabella è facilitata se gli elementi sono disposti seguendo una regola di ordinamento (ad esempio, quello alfabetico). Si supponga di cercare una parola in un dizionario e di aprirlo inizialmente esattamente al centro; si confronta la voce centrale con quella cercata e si stabilisce se quest'ultima si trova nella prima o nella seconda metà del dizionario. Si apre poi al centro la metà del dizionario interessata, si ripete il confronto tra le voci e così via fino a raggiungere la voce desiderata. Se avessimo sfogliato ad una ad una le pagine del dizionario dalla prima fino a quella cercata avremmo quasi sicuramente impiegato più tempo. Il metodo descritto per cercare la parola nel dizionario prende il nome di ricerca binaria. Il principale inconveniente è che l'ordinamento necessario per le chiavi comporta che il metodo di ricerca binaria può essere impiegato solo per tabelle a composizione fissa, in cui in particolare non sia richiesta l'inserzione di nuovi elementi in tempi successivi.
Tutti i problemi che presentano la ricerca completa e la ricerca binaria sarebbero risolti trovando un sistema che permetta di individuare direttamente un elemento e nello stesso tempo di inserire nuovi elementi in ogni momento. Un sistema è quello di calcolare, tramite una funzione (funzione hash), un diverso indirizzo della tabella per ogni elemento che vi si inserisce. La stessa funzione sarà poi usata per la ricerca. Si potranno scegliere funzioni hash di qualsiasi tipo; la loro complessità dipenderà in primo luogo dal numero e dalle caratteristiche degli elementi che la tabella dovrà contenere. Supponiamo ad esempio di voler costruire una tabella contenente informazioni relative a industrie produttrici di automobili e di utilizzare una funzione tale che l'indirizzo di ogni elemento della tabella derivi dalla somma delle posizioni occupate nell'alfabeto dalle prime due lettere della chiave. La chiave FIAT individuerà allora l'indirizzo 15 (F occupa la 6ª posizione nell'alfabeto, I la 9ª, quin-

di l'indirizzo individuato sarà 6+9=15). Se però si sono già introdotte informazioni relative alla OPEL, quando si andranno a inserire i dati della PORSCHE si verificherà una «collisione»: infatti, essendo uguali le prime due lettere che compongono i nomi di queste due case automobilistiche, anche gli indirizzi individuati utilizzando la funzione hash saranno uguali, e quando si cercherà di inserire i dati relativi alla seconda marca si troverà la posizione già occupata. Si può ovviare a questo inconveniente prevedendo di memorizzare le informazioni relative alla PORSCHE nel primo elemento libero che si incontra scorrendo la tabella a partire dall'elemento trovato occupato. Questo procedimento richiede che la tabella sia considerata circolare, cioè che la sua prima posizione sia trattata come consecutiva all'ultima, e comporta inoltre un aumento dei confronti necessari nella ricerca, poiché la ricerca di una chiave procede attraverso gli stessi passi seguiti in fase di inserzione. Nel caso si preveda il verificarsi di molte collisioni, conviene scegliere una funzione più complicata, che riduca al minimo questa eventualità.
Il nostro programma dovrà quindi permettere di svolgere le seguenti operazioni:

- inserire un elemento in tabella
- eliminare un elemento dalla tabella
- visualizzare le informazioni associate ad una chiave
- visualizzare tutti gli elementi della tabella
- memorizzare la tabella su disco
- leggere da disco una tabella precedentemente memorizzata

Per ridurre al minimo la possibilità di collisione, nel calcolo degli indirizzi saranno previste due diverse funzioni hash: se ne userà normalmente una, e si ricorrerà all'altra solo nel caso che la posizione individuata dalla prima risulti già occupata; ciò permetterà di ridurre al minimo l'inserimento degli elementi tramite ricorso allo scorrimento della tabella fino al suo primo indirizzo libero.
Per effettuare la ricerca di una chiave, onde visualizzare le informazioni ad essa associate, il programma dovrà calcolare l'indirizzo individuato dalla chiave fornita utilizzando la prima funzione hash; verificherà poi che la posizione raggiunta contenga effettivamente i dati desiderati, confrontando se la chiave inserita è uguale a quella memorizzata nell'indirizzo individuato.
Se non è così, sarà calcolato un nuovo indirizzo con la seconda funzione e se neppure in questo caso si individuasse l'elemento cercato, si procederà a scorrere la tabella fino a trovare il dato in questione, o a determinare che non esiste un elemento associato alla chiave introdotta.
Per effettuare la cancellazione si userà lo stesso procedimento per accedere all'elemento da eliminare, che sarà poi cancellato rendendo disponibile la posizione che occupava nella tabella per successive inserzioni.
La memorizzazione della tabella su disco avverrà caricando su questo solo gli elementi effettivamente occupati; quando poi si vorrà riutilizzare la tabella memorizzata sul disco si costruirà una tabella vuota in memoria in cui saranno copiati (dal disco) gli elementi contenenti informazioni.
Per visualizzare l'intera tabella si accederà ai suoi elementi in maniera sequenziale, evitando però di stampare gli elementi che risultano liberi.
La prima funzione che sarà adottata per calcolare gli indirizzi eleverà a potenze crescenti i codici ASCII dei caratteri che compongono la chiave fornita, in base alla loro posizione nella chiave stessa, e sommerà poi tra di loro i risultati; la seconda funzione opererà in modo simile, solo che i codici ASCII saranno elevati a potenze decrescenti in base alla loro posizione.

Diagrammi di flusso

Dopo aver inizializzato le costanti e stampato il titolo, il programma chiede se la tabella su cui l'utente intende operare esiste già su disco. Se è così, una volta letto il suo nome, apre il file nel quale è contenuta e la carica in memoria; se si vuole invece creare una nuova tabella è necessario definirla.
In entrambi i casi si confluisce alla visualizzazione del menù in cui sono elencati i comandi predisposti per la gestione (inserzione, eliminazione, visualizzazione di un elemento, visualizzazione e salvataggio di una tabella).
Messo l'utente in condizione di selezionare una scelta, il programma, dopo aver letto il comando, entra nel ciclo che determina quale esecuzione deve associare al comando prescelto.
Al fine di rendere più comprensibile il metodo hash, illustriamo dettagliatamente come si articola il diagramma di inserzione di un elemento nella tabella. Una volta che il programma ha letto la chiave la manipola attraverso una funzione per estrarre da essa un indirizzo.
Quindi si passa a verificare se l'indirizzo ottenuto dalla funzione hash è già occupato. Se è libero si procede all'inserzione dell'elemento, mentre se è occupato ciò può dipendere da due diverse situazioni.

Se l'indirizzo è già occupato dalla medesima chiave (si sta tentando di inserire una chiave già esistente) il programma invierà un messaggio. L'altra possibilità è quella della «collisione»: il valore ottenuto dalla funzione corrisponde all'indirizzo di una chiave precedentemente inserita; in questo caso il programma calcola un altro indirizzo con una nuova funzione hash applicata alla chiave da immettere. Determinato il nuovo indirizzo si controlla se la locazione è libera o no. Qualora sia libera avviene l'inserzione, altrimenti il programma inizia a scorrere in maniera sequenziale la tabella (a partire dalla locazione trovata occupata) e inserisce l'elemento nella prima locazione libera che incontra. Nell'eventualità in cui non si trovi nessuna locazione libera è inviato un messaggio di avvertimento.
Come nel programma «Archivio», anche in questo caso i caratteri «:» e «,» sono interpretati dal driver come comandi, e pertanto non possono essere immessi come dati da memorizzare.
Per la soluzione di questo problema rinviamo al commento e al listato del programma «Editor» (pag. 177), in cui sono inserite due routines che provvedono alla transcodifica prima della memorizzazione e dopo il caricamento da disco.

Il programma

Il programma comincia con l'inizializzazione delle costanti del C-64 (circuito video, circuito suono, memoria video, memoria colore, ecc.) e con la presentazione del titolo (linee 110, 120). Sono poi impostati i caratteri in minuscolo (linea 121), è dimensionata la tabella (linea 147) e sono inizializzate alcune costanti (linea 142). Quindi si inizializza la tavola delle chiavi e dei dati (linee 149÷151). Si chiede poi se la tabella è già memorizzata su disco (linea 170) e si passa il controllo alla routine 60000 che gestisce l'ingresso del carattere digitato in risposta (linea 180). Qualora la tabella si trovi su disco bisogna indicarne il nome (linee 210÷230); il programma apre quindi il canale di comando (linea 250) e quello di comunicazione con il file (linea 260), e controlla che il file contenente la tabella esista (linea 290). In caso di errore è visualizzato un messaggio (linee 310÷360), si chiudono i canali (linee 380, 390) ed è ripresentata la richiesta del nome della tabella (linea 410). Si legge poi la tabella dal disco (linee 430, 440), si controlla se si sono verificati errori (linea 460) e si presenta il menù comandi (linea 560) chiamando la routine 10000. Questa routine gestisce inoltre l'immissione del numero che individua l'operazione desiderata (linea 10110) e

```
0 REM *******************
1 REM *L'ARCHIVIO VELOCE*
2 REM *******************
4 REM VARIABILI UTILIZZATE
6 REM I,J      :CONTATORI DI CICLO
7 REM SV$      :CONTIENE L'IDENTIFICATORE DI STRINGA VUOTA
8 REM VI$      :INDICA CHE LA CHIAVE E' STATA ELIMINATA
9 REM DT       :NUMERO DELLE RIGHE DELLA TABELLA TA$(DT)
10 REM IN$     :INPUT DA TASTIERA
11 REM E,E$    :CODICI DELL'ERRORE SU DISCO
12 REM T,S     :TRACCIA E SETTORE DELL'ERRORE SU DISCO
13 REM I$,TM$:USATE NELLA MEMORIZZAZIONE E RECUPERO DA DISCO
14 REM CO$     :COMANDO IMPOSTATO
15 REM CH$     :CONTIENE LA CHIAVE D'ACCESSO AD UNA RIGA DELLA TABELLA
16 REM TR      :FLAG; SE=1 INDICA TROVATO
17 REM TA$(.):TABELLA CHE CONTIENE LA CHIAVI E LE INFORMAZIONI ASSOCIATE
100 REM TITOLO ED INIZIALIZZAZIONE COSTANTI
101 PRINT CHR$(142):REM CARATTERI MAIUSCOLI
110 PG$="L'ARCHIVIO VELOCE"
120 GOSUB 62000
121 PRINT CHR$(14):REM CARATTERI MINUSCOLI
141 REM INIZIALIZZO COSTANTI USATE
142 SV$="▒▒▒▒▒▒▒▒▒▒":VI$=",,,,,,,,,,":DT=200
146 REM DIMENSIONO LA TAVOLA DELLE CHIAVI E DATI
147 DIM TA$(DT)
148 REM INIZIALIZZO LA TAVOLA DELLE CHIAVI E DATI
149 FOR I=0 TO DT
150 TA$(I)=SV$
151 NEXT I
152 PRINT "■":REM CARATTERI IN NERO
160 REM CHIEDO SE OCCORRE INIZIALIZZARE LA TAVOLA O SE E' GIA' STATA MEMORIZZATA
170 PRINT "♥◙◙LA TABELLA E' SU DISCO ? [S/N] ";
180 ZL=1:GOSUB 60000:PRINT
190 IF IN$<>"S" THEN GOTO 540
200 REM DEVO LEGGERE LA TABELLA DA DISCO
210 PRINT "◙◙/OME DEL FILE ? ";
220 ZL=10:GOSUB 60000:PRINT
230 IF IN$="" THEN PRINT "ΥΥΥ":GOTO 210
240 REM APRO I CANALI DI ERRORE E DATI
250 OPEN 15,8,15
260 OPEN 4,8,4,IN$+",S,R"
270 REM CONTROLLO CHE IL FILE ESISTA
280 INPUT#15,E,E$,T,S
290 IF E<20 THEN GOTO 430
300 REM C'E' UN ERRORE DEL DRIVER
310 PRINT "♥◙◙◙TTENZIONE, ERRORE DEL DRIVER!"
320 PRINT "◙/UMERO     ";E
330 PRINT "◙-ODICE     ";E$
340 PRINT "◙ RACCIA    ";T
350 PRINT "◙◙ETTORE    ";S
360 PRINT "◙◙_ICOMINCIAMO."
370 PRINT#15,"I"
380 CLOSE 4
390 CLOSE 15
400 GOSUB 5000:REM ATTENDO RETURN
410 GOTO 170
420 REM LEGGO LA TABELLA DA DISCO
430 INPUT#4,I$:I=VAL(I$)
431 INPUT#4,TA$(I)
440 IF ST=64 THEN GOTO 471
450 REM CONTROLLO EVENTUALI ERRORI
460 IF ST<>0 AND ST<>64 THEN GOTO 300
470 GOTO 430
471 CLOSE 4:REM CHIUDO CANALE DATI
472 CLOSE 15:REM CHIUDO CANALE COMANDI
540 PRINT "♥"
550 REM PRESENTAZIONE MENU COMANDI
560 GOSUB 10000
570 REM ESEGUO IL COMANDO DATO
580 ON VAL(CO$) GOSUB 20000,21000,22000,23000,23500,24000
590 REM IL COMANDO E' STATO ESEGUITO, RICOMINCIO
630 GOTO560
4990 REM ROUTINE:CHIEDE DI DIGITARE RETURN PER PROSEGUIRE
5000 POKE 214,23:PRINT
5010 PRINT "¬IGITA ▒_¬| /_/▒ PER PROSEGUIRE♥"
5020 GET IN$:IF IN$<>CHR$(13) THEN GOTO 5020
5030 RETURN
9990 REM ROUTINE:PRESENTAZIONE MENU COMANDI
10000 POKE VI+32,6:POKE VI+33,6:REM SFONDO BLU
10001 PRINT "■":REM CARATTERI IN BIANCO
10009 PRINT "♥";TAB(8);"\ ¯ / ,  ¯ Γ \ ♠ / ¯ \"
10010 PRINT "◙1: \NSERISCI UN ELEMENTO IN TABELLA"
10020 PRINT "◙2: ¬LIMINA UN ELEMENTO DALLA TABELLA"
10030 PRINT "◙3: ♦TAMPA LE INFORMAZIONI ASSOCIATE AD     UNA CHIAVE"
10040 PRINT "◙4: ♦ALVA LA TABELLA SU DISCO"
10045 PRINT "◙5: ♦TAMPA TUTTA LA TABELLA"
10050 PRINT "◙6: |ERMINA LA SESSIONE DI LAVORO"
10061 PRINT "◙◙◙¬IGITA LA TUA SCELTA"
10062 PRINT "◙SEGUITA DA ▒_¬| /_/▒ ";
10080 GET IN$
```

l'emissione di un avvertimento sonoro nel caso in cui sia stato digitato un tasto che non è associato ad alcuno dei comandi previsti (linee 10130÷10210). Il controllo torna poi alla linea 580 che chiama la routine che gestisce la funzione richiesta dall'utente.
La routine 20000, che svolge l'operazione di inserimento di un elemento nella tabella, inizia affidando alla routine 40000 il compito di richiedere e di gestire l'immissione della chiave, la quale deve essere comunque pari a dieci caratteri. Qualora infatti se ne fornisca una più corta il programma provvederà ad aggiungervi caratteri che non saranno visualizzati, fino ad arrivare a dieci (linea 40040). Si calcola poi l'indirizzo individuato dalla chiave immessa mediante una prima funzione hash (routines 30000÷30060) e si controlla che non sia già occupato, o che non si tratti di un elemento eliminato precedentemente (linea 20060). Se la posizione risulta già occupata si verifica per prima cosa che non sia stata inserita una chiave già usata (linea 20090), nel qual caso si visualizza un messaggio di avvertimento (linea 20110). Se invece la chiave non è stata usata, ma si è individuata una posizione della tabella occupata da un altro elemento, si calcola una nuova chiave con una seconda funzione (routines 31000÷31060). Infine, qualora risulti occupato anche questo nuovo indirizzo, si cerca una posizione libera scorrendo sequenzialmente la tabella a partire dall'indirizzo

```
10090 IF IN$="" THEN GOTO 10080
10100 REM CONTROLLO CHE IL COMANDO SIA CORRETTO
10110 IF IN$>"0" AND IN$<"7" THEN GOTO 10220
10120 REM IL COMANDO E' ERRATO, AVVISO SONORO
10130 POKE SI+24,15:REM VOLUME
10140 POKE SI,0:POKE SI+1,20:REM FREQUENZA
10150 POKE SI+5,15:REM ATTACK,DECAY
10160 POKE SI+6,240:REM SUSTAIN, RELEASE
10170 POKE SI+4,17:REM FORMA D'ONDA
10180 FOR I=1 TO 80:NEXT I:REM RITARDO
10190 POKE SI+4,0:REM DISATTIVO EMISSIONE SONORA
10200 POKE SI+24,0:REM VOLUME A ZERO
10210 GOTO 10080
10220 PRINT "▒";IN$;"▒";
10221 CO$=IN$
10230 GET IN$
10240 IF IN$=CHR$(20) THEN PRINT IN$;:GOTO 10080:REM CARATTERE CMD
10250 IF IN$<>CHR$(13) THEN GOTO 10230
10260 RETURN
19990 REM ROUTINE:INSERISCE UN ELEMENTO IN TABELLA
20000 PRINT "▒";TAB(15);"▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
20010 REM RICHIEDO CHIAVE D'ACCESSO
20020 GOSUB 40000:IF CH$="" THEN RETURN
20030 REM CALCOLO INDIRIZZO TRAMITE UNA PRIMA FUNZIONE HASH
20040 GOSUB 30000
20050 REM CONTROLLO CHE IL POSTO NON SIA GIA' OCCUPATO
20051 REM OPPURE CHE L'ELEMENTO CORRISPONDENTE NON SIA STATO ELIMINATO
20060 IF (LEFT$(TA$(PO),10)=SV$) OR (LEFT$(TA$(PO),10)=VI$) THEN GOTO 20330
20070 REM IL POSTO E' GIA' OCCUPATO
20080 REM CONTROLLO CHE NON SIA UN ELEMENTO CON LA STESSA CHIAVE D'ACCESSO
20090 IF LEFT$(TA$(PO),10)<>CH$ THEN GOTO 20170
20100 REM LA CHIAVE E' LA STESSA
20110 PRINT "▒▒▒TTENZIONE, LA CHIAVE ▒";CH$;"▒ E'"
20120 PRINT "▒GIA' STATA USATA."
20130 GOSUB 5000:REM ATTENDO RETURN
20140 REM RICOMINCIO
20150 RETURN
20160 REM PROVO CON UNA SECONDA FUNZIONE HASH
20170 GOSUB 31000
20180 REM CONTROLLO CHE IL POSTO NON SIA GIA' OCCUPATO
20190 IF (LEFT$(TA$(PO),10)=SV$) OR (LEFT$(TA$(PO),10)=VI$) THEN GOTO 20330
20210 REM CONTROLLO CHE NON SIA UN ELEMENTO CON LA STESSA CHIAVE D'ACCESSO
20220 IF LEFT$(TA$(PO),10)<>CH$ THEN GOTO 20260
20230 REM LA CHIAVE E' LA STESSA
20240 GOTO 20110
20250 REM CERCO SEQUENZIALMENTE I PRIMO POSTO LIBERO O UNA CHIAVE UGUALE
20260 I=PO+1
20270 IF I=DT+1 THEN I=0
20280 IF I=PO THEN GOTO 20410:REM LA TABELLA E' PIENA
20290 IF (LEFT$(TA$(I),10)=SV$) OR (LEFT$(TS$(I),10)=VI$) THEN PO=I:GOTO 20330
20300 IF LEFT$(TA$(I),10)=CH$ THEN GOTO 20110:REM LA CHIAVE E' GIA' STATA USATA
20310 I=I+1:GOTO 20270
20320 REM E' STATO TROVATO UN POSTO LIBERO
20330 PRINT "▒▒▒IGITA L'INFORMAZIONE ASSOCIATA ALLA"
20340 PRINT "▒CHIAVE ▒";CH$;"▒ , HAI A DISPOSIZIONE"
20350 PRINT "▒78 CARATTERI."
20360 PRINT "▒▒NFORMAZIONE ASSOCIATA:"
20361 PRINT "▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
20362 PRINT
20363 ZL=78:GOSUB 60000:PRINT
20364 IF IN$="" THEN PRINT "▒▒▒▒▒▒▒":GOTO 20360
20370 REM MEMORIZZO L'INFORMAZIONE
20380 TA$(PO)=CH$+IN$
20390 REM HO TERMINATO L'INSERZIONE
20400 RETURN
20410 REM LA TABELLA E' COMPLETAMENTE RIEMPITA
20420 PRINT "▒▒▒▒▒▒▒TTENZIONE, LA TABELLA E' PIENA."
20430 PRINT "▒▒ON POSSONO ESSERE EFFETTUATE INSERZIONI"
20440 GOSUB 5000:REM ATTENDO RETURN
20450 RETURN
20990 REM ROUTINE:ELIMINA UN ELEMENTO DALLA TABELLA
21000 PRINT "▒";TAB(14);"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
21010 REM RICHIEDO CHIAVE D'ACCESSO
21020 GOSUB 40000:IF CH$="" THEN RETURN
21030 REM CERCO LA CHIAVE CH$ IN TABELLA
21040 GOSUB 32000
21060 IF TR=1 THEN GOTO 21130
21070 REM LA CHIAVE NON E' STATA TROVATA
21080 PRINT "▒▒▒TTENZIONE, LA CHIAVE ▒";CH$;"▒ NON"
21090 PRINT "▒E' PRESENTE IN TABELLA."
21100 GOSUB 5000:REM ATTENDO RETURN
21110 RETURN
21120 REM LA CHIAVE E' STATA TROVATA, LA ELIMINO
21130 TA$(PO)=VI$
21140 PRINT "▒LA CHIAVE ▒";CH$;"▒ E' STATA"
21150 PRINT "▒ELIMINATA DALLA TABELLA."
21160 GOSUB 5000:REM ATTENDO RETURN
21170 RETURN
21990 REM ROUTINE:RICERCA UN ELEMENTO IN TABELLA
22000 PRINT "▒";TAB(17);"▒▒▒▒▒▒▒▒▒▒▒"
```

immediatamente successivo a quello trovato occupato fino a tornare alla posizione individuata da quest'ultimo (linee 20270÷20310). Se non si trova alcun posto libero la tabella è ovviamente piena, si visualizza allora un messaggio (linee 20420, 20430) e si ripresenta il menù comandi. Quando invece si è individuata una posizione libera, si richiede di inserire i dati associati alla chiave (linee 20330÷20360), che è quindi memorizzata insieme a questi (linea 20370), per poi ripresentare il menù comandi.
La cancellazione di un elemento della tabella è gestita dalla routine 21000, che chiede la chiave dell'elemento che si vuole eliminare (linea 21020) e passa il controllo alla routine 32000. Questa routine, calcolato l'indirizzo individuato dalla chiave con la prima funzione (linea 32020), controlla se a tale indirizzo corrisponde realmente l'elemento cercato (linea 32030) o se non vi corrisponde alcun elemento. Qualora non si verifichi uno di questi casi significa che è avvenuta una collisione; è allora necessario calcolare un nuovo indirizzo con la seconda funzione (linea 32060). Se anche la nuova posizione non contiene l'elemento desiderato e non è vuota (linee 32070, 32080) i dati richiesti si cercano scorrendo sequenzialmente la tabella (linee 32120÷32160). Il controllo è poi restituito alla linea 21060, dove troviamo il test per verificare se è stato individuato l'elemento cercato. Nel caso in cui l'elemento non sia stato trovato si informa l'utente

```
22010 REM RICHIEDO CHIAVE D'ACCESSO
22020 GOSUB 40000:IF CH$="" THEN RETURN
22030 REM CERCO LA CHIAVE CH$ IN TABELLA
22040 GOSUB 32000
22060 IF TR=1 THEN GOTO 22130
22070 REM LA CHIAVE NON E' STATA TROVATA
22080 PRINT "██▲TTENZIONE, LA CHIAVE █";CH$;"█ NON"
22090 PRINT "█E' PRESENTE IN TABELLA."
22100 GOSUB 5000:REM ATTENDO RETURN
22110 RETURN
22120 REM LA CHIAVE E' STATA TROVATA, STAMPO L'INFORMAZIONE ASSOCIATA
22130 PRINT "█▚NFORMAZIONE ASSOCIATA:"
22140 PRINT "▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔"
22150 PRINT "█";RIGHT$(TA$(PO),LEN(TA$(PO))-10)
22160 GOSUB 5000:REM ATTENDO RETURN
22170 RETURN
22990 REM ROUTINE:SALVA LA TABELLA SU DISCO
23000 PRINT "█";TAB(10);"█▲▲L╳▲|▲||╲┌ ♥╱ ┐╲♥┬┌████"
23010 PRINT "█╱OME DEL FILE ? ";
23020 ZL=10:GOSUB 60000:PRINT
23030 IF IN$="" THEN RETURN
23031 REM APRO IL FILE COMANDI
23032 OPEN 15,8,15
23040 REM APRO IL FILE IN MODO SOVRAPPOSIZIONE
23050 OPEN 4,8,4,"@0:"+IN$+",S,W"
23051 REM CONTROLLO ERRORI
23052 INPUT#15,E,E$,T,S
23053 IF E<20 THEN GOTO 23070
23054 REM AVVISO ERRORE
23055 PRINT "██▲TTENZIONE, ERRORE DEL DISCO"
23056 PRINT "╱UMERO    :";E
23057 PRINT "━ODICE    :";E$
23058 PRINT "|RACCIA   :";T
23059 PRINT "♥ETTORE   :";S
23060 GOSUB 5000:REM ATTENDO RETURN
23061 CLOSE 4
23062 PRINT#15,"I"
23063 CLOSE 15
23064 GOTO 23000
23069 REM MEMORIZZO SUL FILE IN$
23070 FOR I=0 TO DT
23080 IF (LEFT$(TA$(I),10)=SV$) OR (LEFT$(TA$(I),10)=VI$) THEN GOTO 23120
23090 PRINT#4,STR$(I)
23091 PRINT#4,TA$(I)
23120 NEXT I
23130 CLOSE 4
23131 CLOSE 15
23140 PRINT "██LA TABELLA E' STATA MEMORIZZATA."
23150 GOSUB 5000:REM ATTENDO RETURN
23160 RETURN
23490 REM ROUTINE:STAMPA TUTTA LA TABELLA
23500 PRINT "█";TAB(13);"♥|▲╲┐▲ |▲|┐┴▲"
23510 FOR I=0 TO DT
23520 TM$=LEFT$(TA$(I),10)
23530 IF TM$=SV$ OR TM$=VI$ THEN GOTO 23610
23540 REM L'ELEMENTO I-ESIMO ESISTE, LO STAMPO
23541 PRINT "█";TAB(13);"♥|▲╲┐▲ |▲|┐┴▲"
23550 POKE 214,10:PRINT
23560 PRINT "━HIAVE: █";TM$;"█"
23570 PRINT "█▚NFORMAZIONE ASSOCIATA:"
23580 PRINT "▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔"
23590 PRINT "██";RIGHT$(TA$(I),LEN(TA$(I))-10)
23600 GOSUB 5000:REM ATTENDO UN RETURN
23610 NEXT I
23620 RETURN
24000 PRINT "█";TAB(13);"_╲╱▔ ♥▔♥♥╲┌╱▔███"
24010 PRINT "██╳UOI VERAMENTE TERMINARE ? [S/N] ";
24020 ZL=1:GOSUB 60000:PRINT
24030 IF IN$="S" THEN PRINT "███";CHR$(142)
24035 IF IN$="S" THEN POKE53280,14:PRINT"██":END
24040 RETURN
29980 REM ROUTINE:CALCOLA L'INDIRIZZO ASSOCIATO ALLA CHIAVE CH$ TRAMITE
29990.REM L'USO DELLA FUNZIONE HASH NUMERO UNO
30000 PO=0
30010 FOR I=1 TO 10
30020 PO=PO+ASC(MID$(CH$,I,1))↑((I-1)/3+1)
30030 PO=PO-INT(PO/(DT+1))*(DT+1)
30040 NEXT I
30060 RETURN
30980 REM ROUTINE:CALCOLA L'INDIRIZZO ASSOCIATO ALLA CHIAVE CH$ TRAMITE
30990 REM L'USO DELLA FUNZIONE HASH NUMERO DUE
31000 PO=0
31010 FOR I=1 TO 10
31020 PO=PO+ASC(MID$(CH$,11-I,1))↑((I-1)/4+1)
31030 PO=PO-INT(PO/(DT+1))*(DT+1)
31040 NEXT I
31060 RETURN
31990 REM ROUTINE:CERCA LA CHIAVE CH$ IN TABELLA
32000 TR=0:REM FLAG TROVATO = FALSO
32010 REM CALCOLO L'INDIRIZZO TRAMITE LA PRIMA FUNZIONE HASH
```

che la chiave fornita non è presente nella tabella (linee 21080, 21090), altrimenti si cancella l'elemento (linea 21130) e si ripresenta il menù comandi.
La routine 22000 ricerca e visualizza un elemento della tabella. Chiesta al solito la chiave (linea 22020), il controllo è passato alla routine 32000, che gestisce l'individuazione dell'elemento associato alla chiave fornita. Se la ricerca ha buon esito si stampa l'informazione contenuta nella posizione individuata (linee 21130÷21150), in caso contrario si informa l'utente che la chiave fornita non è presente nella tabella (linee 22080, 22090).
La memorizzazione della tabella su disco è eseguita dalla routine 23000 che, chiesto il nome che si vuole assegnare alla tabella (linea 23010), apre un file sul disco (anche se questo è già presente) specificando che vi si deve scrivere sopra (linea 23050). Si memorizzano quindi tutti gli elementi della tabella che contengono informazioni, per poi chiudere il file (linea 23130) e ripresentare il menù comandi.
La routine 23500 visualizza l'intera tabella. Prima di stampare un elemento (linee 23560÷23590) si controlla se contiene o no informazioni (linea 23530) per evitare di visualizzare gli elementi vuoti.
La routine 24000, infine, termina la sessione di lavoro dopo aver chiesto se si ha veramente intenzione di finire di operare sulla tabella (linea 24010). Si permette così di riprendere l'esecuzione con la presentazione del menù comandi.

```
32020 GOSUB 30000
32030 IF LEFT$(TA$(PO),10)=CH$ THEN TR=1:RETURN:REM CHIAVE TROVATA
32040 IF LEFT$(TA$(PO),10)=SV$ THEN RETURN:REM CHIAVE NON TROVATA
32050 REM CALCOLO L'INDIRIZZO CON LA SECONDA FUNZIONE HASH
32060 GOSUB 31000
32070 IF LEFT$(TA$(PO),10)=CH$ THEN TR=1:RETURN
32080 IF LEFT$(TA$(PO),10)=SV$ THEN RETURN
32090 REM CERCO LA CHIAVE IN SEQUENZIALE
32110 I=PO+1
32120 IF I=DT+1 THEN I=0
32130 IF I=PO THEN RETURN
32140 IF LEFT$(TA$(I),10)=SV$ THEN RETURN
32150 IF LEFT$(TA$(I),10)=CH$ THEN PO=I:TR=1:RETURN
32160 I=I+1:GOTO 32120
39990 REM ROUTINE:RICHIEDE CHIAVE D'ACCESSO
40000 PRINT "-HIAVE D'ACCESSO ? ";
40010 ZL=10:GOSUB 60000:PRINT
40011 CH$=IN$
40020 IF CH$="" THEN RETURN
40030 REM LA LUNGHEZZA DELLA CHIAVE DEVE ESSERE PARI A 10 CARATTERI
40040 IF LEN(CH$)<10 THEN CH$="▒"+CH$:GOTO 40040
40060 RETURN
59920 REM ROUTINE:GESTISCE L'INPUT DI UNA STRINGA ALFANUMERICA
59940 REM IN INGRESSO VUOLE IL VALORE ZL CHE E' LA LUNGHEZZA MASSIMA DELLA
59950 REM STRINGA DA LEGGERE
59960 REM IN USCITA DA' LA STRINGA LETTA IN IN$ E LA SUA LUNGHEZZA IN Z9
59980 REM CANCELLO LA ZONA DI SCHERMO IN CUI VERRA' DIGITATA LA STRINGA
60000 FOR Z8=1 TO ZL: PRINT " ";:NEXT Z8
60010 FOR Z8=1 TO ZL: PRINT "▒";:NEXT Z8
60020 IN$="":Z7=TI
60040 REM LEGGO UN CARATTERE
60060 GET Z8$:IF Z8$<>"" THEN 60160
60080 REM ACCENSIONE E SPEGNIMENTO DEL CURSORE
60100 IF Z7<TI AND NOT(Z6) THEN PRINT "▒▒";:Z6=NOT(Z6):Z7=TI+15
60110 IF Z7<TI AND Z6 THEN PRINT " ▒";:Z6=NOT(Z6):Z7=TI+15
60120 GOTO 60060
60140 REM E' STATO DIGITATO UN CARATTERE
60160 Z8=ASC(Z8$):Z9=LEN(IN$)
60180 REM SE NON E' UN CARATTERE ALFANUMERICO, DEVE ESSERE UN RETURN O DELETE
60200 IF NOT(Z8>31 AND Z8<96) THEN GOTO 60320
60220 REM CONTROLLO CHE NON SIA STATA SUPERATA LA LUNGHEZZA MASSIMA
60240 IF Z9=ZL THEN GOTO 60060
60260 REM LO AGGIUNGO ALLA STRINGA IN$
60280 IN$=IN$+Z8$:PRINT Z8$;:GOTO 60060
60300 REM SE E' UN RETURN, HO TERMINATO LA LETTURA
60320 IF Z8=13 THEN PRINT " ▒";:RETURN
60340 REM SE E' DELETE, CANCELLO IN IN$ E SUL VIDEO L' ULTIMO CARATTERE DIGITATO
60360 IF Z8=20 AND Z9>0 THEN IN$=LEFT$(IN$,Z9-1):PRINT " ▒▒";:GOTO 60060
60370 GOTO 60060
60960 REM ROUTINE:INIZIALIZZAZIONE COSTANTI
60980 REM CIRCUITO VIDEO
61000 VI=53248
61020 REM CIRCUITO SUONO
61040 SI=54272
61060 REM MEMORIA VIDEO
61080 MV=1024
61100 REM MEMORIA COLORE
61120 MC=55296
61140 REM COSTANTI DI USO COMUNE
61160 ZL=9
61180 REM INIZIALIZZAZIONE CHIP SUONO
61200 FOR I=0 TO 24
61210 POKE SI+I,0
61220 NEXT I
61230 RETURN
61980 REM ROUTINE:STAMPA IL TITOLO DEL PROGRAMMA MEMORIZZATO IN PG$
62000 GOSUB 61000
62010 POKE VI+32,15
62020 POKE VI+33,15
62030 PRINT "▒▒▒▒▒";
62040 PRINT TAB(6);"┌────────────────────────────┐"
62050 FOR I=1 TO 5
62060 PRINT TAB(6);"│                            │"
62070 NEXT I
62080 PRINT TAB(6);"└────────────────────────────┘"
62090 PRINT TAB(6);"▒▒▒▒▒▒▒DIGITA ▒RETURN▒ PER PROSEGUIRE"
62100 PRINT "▒▒▒▒"
62110 FOR I=1 TO 5
62120 PRINT TAB(7);
62130 FOR J=1 TO 26
62140 PRINT "▒▒ ";
62150 NEXT J
62160 PRINT
62170 NEXT I
62190 REM ORA SCRIVO IL TITOLO
62210 PRINT "▒▒▒▒▒▒▒▒";TAB((40-LEN(PG$))/2),"▒▒";PG$
62220 GET Z9$
62230 IF Z9$<>CHR$(13) THEN GOTO 62220
62240 PRINT "▒▒";
62250 RETURN
```

Editor

Questo progetto ha come scopo la realizzazione di un programma che permetta di comporre un testo qualsiasi ed eventualmente di memorizzarlo in un file («text file»). Si potrà, ad esempio, scrivere una lettera utilizzando il calcolatore in sostituzione della macchina per scrivere. Durante la composizione del testo il programma permetterà di scrivere righe, di effettuare inserzioni tra righe già scritte, di apportare modifiche, di cercare e/o sostituire una determinata parola, ecc. Si potranno così comprendere ed apprezzare le ampie possibilità di «word processing» (elaborazione della parola) proprie dei computer in contrasto con le limitazioni imposte nell'uso delle comuni macchine per scrivere. Inoltre, potendo memorizzare il testo su supporto magnetico, lo si avrà a disposizione per successivi impieghi e/o modifiche.

Analisi del problema

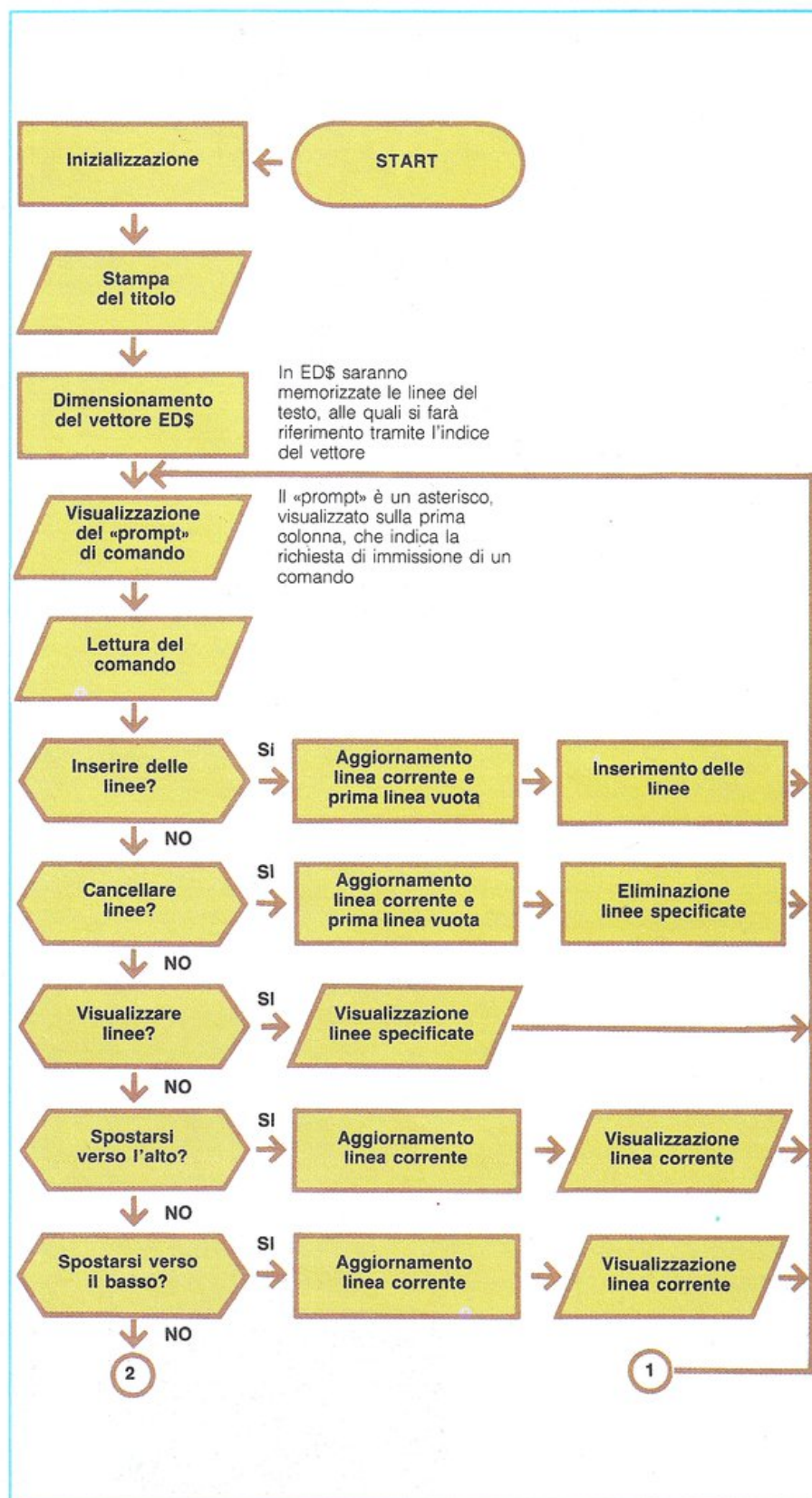
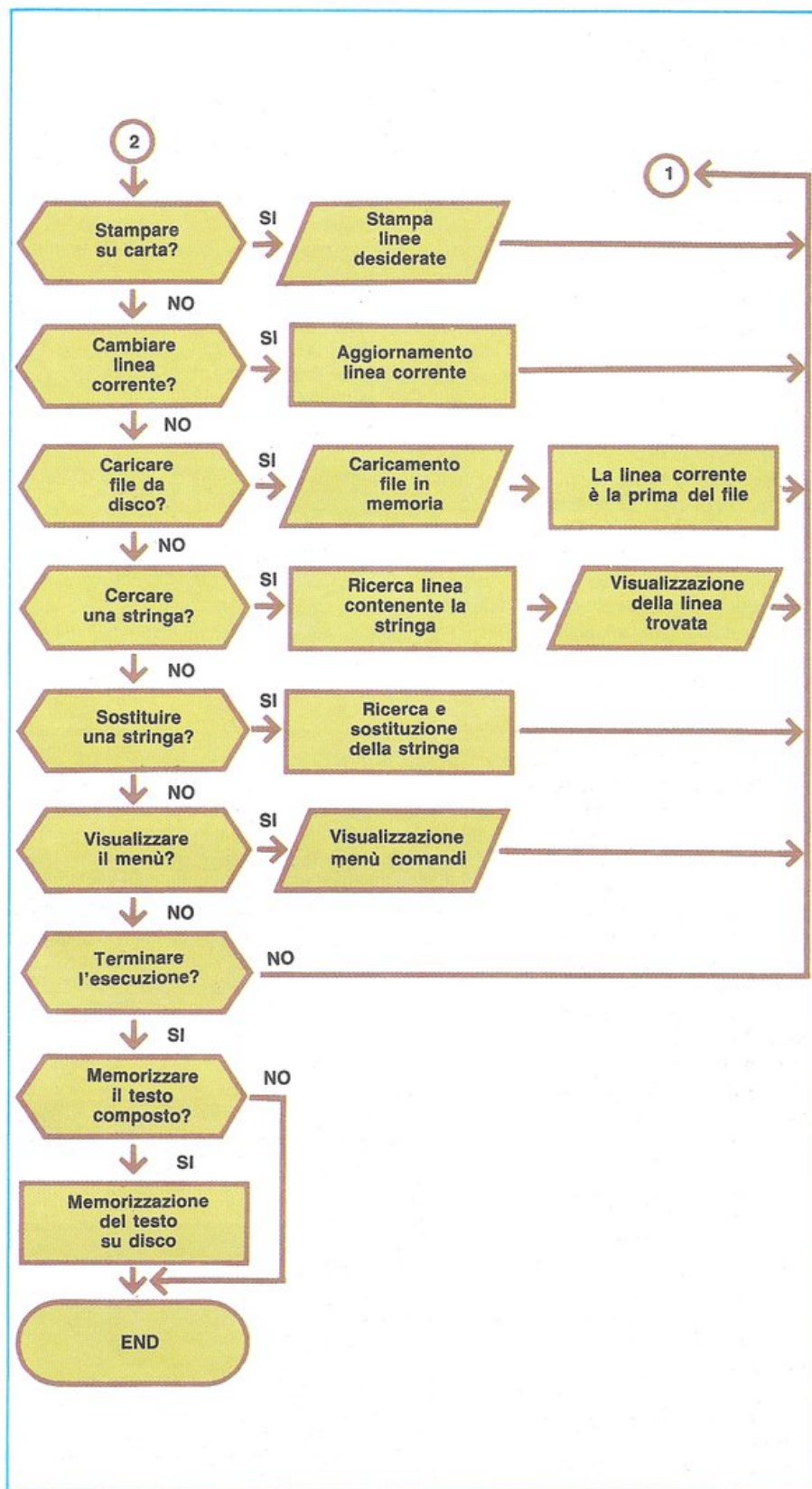
Un programma che permette di creare e manipolare testi in modo semplice e di memorizzarli su supporto magnetico per poterli utilizzare successivamente è generalmente chiamato «editor». Esistono due famiglie di editor: gli editor orientati alla linea e quelli orientati alla pagina. Gli editor orientati alla linea manipolano il testo linea per linea, cioè lo considerano come una successione di linee. Prevedono infatti anche un comando che consente di definire una «linea corrente» cioè la linea su cui si possono svolgere operazioni. Il loro limite principale è che non è possibile operare su una linea (correggere errori, cambiare caratteri, ecc.) se non richiamandola espressamente. Gli editor orientati alla pagina invece trattano il testo come una successione di gruppi di linee (pagine) e consentono di operare in un campo più vasto spostandosi sulle linee che compongono la pagina. Poiché la loro gestione è molto più complessa, il progetto che vogliamo realizzare è quello di un editor orientato alla linea. Le operazioni che il programma dovrà essere in grado di eseguire saranno le seguenti:

- inserimento delle linee del testo
- visualizzazione di una o più linee di testo
- spostamento della linea corrente verso la fine del testo
- spostamento della linea corrente verso l'inizio del testo
- individuazione di una parola o di un qualsiasi insieme di caratteri (stringa) specificato e visualizzazione della linea in cui si trova
- visualizzazione di un quadro contenente la spiegazione dei comandi a disposizione
- individuazione di una stringa nella linea corrente o nelle successive, e sua sostituzione con un'altra stringa
- selezione della linea corrente
- eliminazione di una o più linee di testo
- stampa del testo o di una sua parte su carta
- caricamento di un testo da supporto magnetico
- fine della sessione di lavoro con memorizzazione del testo su supporto magnetico
- fine della sessione di lavoro con distruzione del testo.

Ai comandi che individuano queste operazioni saranno associati alcuni parametri; sarà previsto che il comando sia rappresentato da un carattere e che gli eventuali caratteri successivi individuino i parametri associati (cioè le modalità d'uso del comando). In particolare i comandi: «i» (insert: inserisce delle linee), «f» (find: cerca una stringa), «h» (help: visualizza i comandi), «s» (substitute: sostituisce una stringa), «m» (move: carica un testo da disco), «e» (exit: termina la sessione di lavoro memorizzando il testo su disco) e «q» (quit: termina la sessione di lavoro di-

struggendo il testo) non avranno parametri associati, in quanto possono essere utilizzati in modo univoco. I comandi: «l» (list: visualizza una o più linee), «k» (kill: elimina una o più linee) e «p» (print: stampa una o più linee) possono invece essere utilizzati associandovi alcuni parametri. Ad esempio, il comando «l» visualizza la linea corrente, «l –» visualizza tutte le linee a partire da quella corrente fino alla fine del testo, «l4» visualizza la linea 4 anche se quella corrente è un'altra e «l10 – 20» visualizza le linee dalla 10ª alla 20ª. Al comando «c» (change: seleziona la linea corrente) deve essere associato il numero della linea da selezionare e ai comandi «u» (up: sposta la linea corrente verso l'inizio del testo) e «d» (down: sposta la linea corrente verso la fine del testo) possono essere associati numeri che indicano di quante linee si vuole che sia effettuato lo spostamento. Per individuare gli eventuali parametri associati ad un comando, dopo aver tolto i blanks presenti, sarà estratto, dalla stringa immessa dall'utente per chiedere l'esecuzione del comando, il primo carattere (che individua il comando), e saranno poi controllati i successivi per vedere se si tratta di parametri.
Generalmente la progettazione di un editor è in realtà più complessa; si possono infatti prevedere altre funzioni, quali lo spostamento di gruppi di linee all'interno del testo, la fusione di due o più testi tra di loro, ecc. Il testo è inoltre memorizzato su supporto magnetico non al termine della sessione di lavoro, come in questo caso, ma durante la sua composizione, per evitare che possa andare completamente perduto nell'eventualità, ad esempio, di un'improvvisa mancanza di corrente elettrica.
Parleremo ora del problema che i lettori più attenti non avranno mancato di rilevare nei programmi «Archivio» e «L'archivio veloce», che al pari di «Editor» utilizzano estesamente il trasferimento di dati alfabetici al disco.
Il trasferimento di una stringa di caratteri presenta qualche problema quando in essa sono presenti i caratteri «:» e «,», che il driver interpreta come comandi. In questo programma si è utilizzata una soluzione che prevede la transcodifica dei caratteri citati. Prima del trasferimento al driver i caratteri «:» e «,» sono sostituiti con altri «innocui» ([e]), che all'atto della lettura dal disco sono ritradotti nei caratteri effettivamente immessi dall'utente. È chiaro però che in questo modo non sarà più possibile utilizzare nel testo le parentesi quadre, che sarebbero comunque tradotte nei caratteri «:» e «,». Un secondo problema riguarda invece l'uso dei doppi apici. All'atto dell'immissione di un doppio apice il CBM-64 entra in una modalità di funzionamento particolare, che cessa solo con l'immissione degli apici di chiusura. Se il programma stampasse questo carattere senza ulteriori verifiche, le linee 60100 e 60110 sarebbero interpretate nel seguito come istruzioni di stampa dei caratteri grafici contenuti fra i doppi apici. Per evitare questo inconveniente il programma stamperà i doppi apici per due volte consecutive nella stessa posizione, permettendo così l'immissione di questo carattere nel testo.

Diagrammi di flusso

Dopo aver inizializzato le costanti e stampato il titolo, si dimensiona il vettore ED$ in cui verranno collocate le righe del testo. La visualizzazione di un asterisco indica che il programma è in attesa di un comando. Dopo la lettura del comando digitato si entra in un ciclo di individuazione dell'azione che vi corrisponde. Se si intende scrivere delle linee di testo (comando «i»), inserendole di seguito alla linea corrente, sono innanzitutto aggiornati i contatori relativi alla riga corrente e alla prima riga vuota, quindi si accettano le nuove righe di scrittura inserendole nelle relative posizioni (rispetto al testo). Non appena si tenta di inserire una linea vuota il programma la ignora e ritorna ad attendere un comando uscendo dalla modalità di inserimento. Nel caso in cui si vogliano cancellare delle linee è necessario digitare il comando «k» seguito dai relativi numeri di riga. Per leggere il testo composto si deve digitare «l» specificando le righe da visualizzare, che saranno stampate sullo schermo. I blocchi che seguono verificano se si intende spostare la linea corrente in alto o in basso (comandi «u» e «d»); in entrambi i casi si aggiorna il relativo puntatore e si visualizza la nuova linea. È possibile stampare su carta il testo o una sua parte con il comando «p»;

il programma gestisce l'apertura e la chiusura dei canali di comunicazione con la stampante, e produce la stampa (su carta) delle righe desiderate.
Per cambiare la linea corrente bisogna digitare «c» seguito dal nuovo numero di riga.
Nel blocco successivo si effettua il test relativo al comando «m» (caricare un file da disco); se si tratta di questo comando il testo è prelevato da disco e memorizzato nel vettore ED$ (la prima linea del testo sarà quella corrente).
Nel seguito si incontrano i test relativi alla ricerca e alla sostituzione di una stringa (comandi «f» e «s»).
In entrambi i casi si ricerca la stringa all'interno del testo; quando la si individua è visualizzata la linea che la contiene (ricerca) ed eventualmente la vecchia stringa è sostituita con la nuova (sostituzione).
Il comando «h» richiede la visualizzazione del menù dei comandi, e in seguito è necessario digitare RETURN perché il programma torni ad attendere una nuova istruzione.
Gli ultimi blocchi del diagramma verificano se si intende terminare la composizione del testo (e l'esecuzione). Il testo finale può essere memorizzato su disco prima di terminare l'esecuzione (comando «e») oppure si può concludere senza memorizzarlo e quindi perdendolo (comando «q»).

Il programma

Il programma comincia con l'inizializzazione delle costanti del C-64 e con la presentazione del titolo (linee 110, 120). Alla linea 140 è inizializzata la costante DW che serve a determinare le dimensioni del vettore ED$, il quale conterrà le linee del testo (linea 160). Sono poi impostati i caratteri in minuscolo (linea 280) ed è disabilitata la possibilità di cambiare questa impostazione (linea 290). Si determinano quindi i colori del video (linee 300, 310) e dei caratteri (linea 320), e si inizializzano le costanti LC e FT che individuano rispettivamente la linea corrente e la prima linea vuota (linee 340, 341). Con la linea 370 si visualizza il carattere «stella» per indicare che si possono utilizzare i comandi dell'editor, e con la linea 380 è chiamata la routine 60000 che gestisce l'immissione dei caratteri da tastiera. Si passa poi il controllo alla routine 41000, che toglie gli eventuali spazi bianchi immessi (linee 41010÷41040) ed estrae il primo carattere della stringa mediante la funzione LEFT$ (linea 410). Le linee 420÷540 effettuano la chiamata alla routine che gestisce il comando scelto.
La routine 10000, che effettua l'inserimento delle linee del testo, passato il controllo alla routine 60000 per gestire l'immissione dei caratteri che compongono la linea, verifica se è stato battuto il tasto RETURN

```
0 REM ********
1 REM *EDITOR*
2 REM ********
4 REM VARIABILI UTILIZZATE
6 REM PG$          :REM MEMORIZZA IL TITOLO DEL PROGRAMMA
7 REM I,J          :CONTATORI DI CICLO
8 REM DW           :MASSIMO NUMERO DI RIGHE DEL TESTO
9 REM LC           :NUMERO DELLA LINEA CORRENTE
10 REM FT          :NUMERO DELLA PRIMA LINEA VUOTA
11 REM IN$         :INPUT DA TASTIERA
12 REM CO$         :COMANDO IMMESSO
13 REM IN          :PRIMA LINEA DEL BLOCCO DA LISTARE OD ELIMINARE
14 REM FI          :ULTIMA LINEA DEL BLOCCO DA LISTARE OD ELIMINARE
15 REM CI$         :VALORE TEMPORANEO
16 REM TM$         :SERVE NELLA COSTRUZIONE DI CI$
17 REM I$          :STRINGA CORRISPONDENTE AL NUMERO DI LINEA DA STAMPARE
18 REM N           :INDICA DI QUANTE LINEE CI SI DEVE SPOSTARE
19 REM E,E$,T,S    :CODICI DELL'ERRORE SU DISCO
20 REM FG          :VARIABILE BOOLEANA
21 REM BU$         :VALORE TEMPORANEO
22 REM ED$(.)      :VETTORE ALFANUMERICO CHE CONTIENE IL TESTO
100 REM TITOLO ED INIZIALIZZAZIONE COSTANTI C64
110 PG$="EDITOR"
120 GOSUB 62000
130 REM INIZIALIZZO VARIABILI
140 DW=300:REM MASSIMO NUMERO DI LINEE DELL'EDITOR
150 REM INIZIALIZZO ARRAY
160 DIM ED$(DW)
162 PRINT "▓"
164 PRINT-SPC(5);"INIZIALMENTE L'EDITOR E'"
166 PRINT SPC(5);"▓IN STATO COMANDI."
168 PRINT SPC(5);"▓PER SCRIVERE UN TESTO"
170 PRINT SPC(5);"▓E' NECESSARIO DIGITARE"
172 PRINT SPC(5);"▓'I' SEGUITO DA ▓RETURN■."
174 PRINT SPC(5);"▓SI RITORNA IN STATO COMANDI"
176 PRINT SPC(5);"▓DIGITANDO ▓RETURN■ SU"
178 PRINT SPC(5);"▓UNA LINEA VUOTA."
180 PRINT SPC(5);"▓PER LEGGERE IL MENU'"
182 PRINT SPC(5);"▓DIGITARE 'H' IN STATO COMANDI."
184 PRINT "▓▓▓▓RETURN■ PER PROSEGUIRE"
186 GET A$:IF A$<>CHR$(13) THEN 186
188 PRINT"▓"
280 PRINT CHR$(14):REM CARATTERI MINUSCOLI
290 POKE 657,128:REM DISABILITO IL CAMBIO DI CARATTERI
300 POKE VI+32,6:REM CORNICE BLU
310 POKE VI+33,6:REM SFONDO BLU
320 PRINT "▓":REM CARATTERI BIANCHI
330 PRINT "▓":REM CLEAR VIDEO
340 LC=0:REM LA LINEA CORRENTE E' LA ZERO
341 FT=0:REM LA PRIMA LINEA NON SCRITTA E' LA ZERO
350 REM INIZIA IL PROGRAMMA
360 REM SONO IN MODO COMANDO
370 PRINT "*";:REM PROMPT DEL MODO COMANDO
380 ZL=70:GOSUB 60000:PRINT:REM LEGGO STRINGA DI COMANDO
390 REM PROCESSO LA STRINGA DI COMANDO
391 GOSUB 41000:REM TOLGO GLI SPAZI BIANCHI IN IN$
400 IF IN$="" THEN GOTO 370
410 CO$=LEFT$(IN$,1)
420 IF CO$="I" THEN GOSUB 10000:GOTO 370:REM INSERT
430 IF CO$="K" THEN GOSUB 11000:GOTO 370:REM CANCELLA LINEE
440 IF CO$="L" THEN GOSUB 12000:GOTO 370:REM LISTA
450 IF CO$="Q" THEN GOSUB 13000:GOTO 370:REM QUIT
460 IF CO$="E" THEN GOSUB 14000:GOTO 370:REM EXIT
470 IF CO$="U" THEN GOSUB 15000:GOTO 370:REM IN ALTO DI N LINEE
480 IF CO$="D" THEN GOSUB 16000:GOTO 370:REM IN BASSO DI N LINEE
490 IF CO$="P" THEN GOSUB 17000:GOTO 370:REM LISTA SU STAMPANTE
500 IF CO$="C" THEN GOSUB 18000:GOTO 370:REM CAMBIA IL NUMERO DI LINEA CORRENTE
510 IF CO$="M" THEN GOSUB 19000:GOTO 370:REM LEGGE UN FILE DA DISCO
520 IF CO$="F" THEN GOSUB 20000:GOTO 370:REM CERCA UNA STRINGA
530 IF CO$="S" THEN GOSUB 21000:GOTO 370:REM SOSTITUISCE UNA STRINGA
540 IF CO$="H" THEN GOSUB 22000:GOTO 370:REM HELP
1000 GOTO370
9990 REM ROUTINE:GESTISCE L'INSERZIONE SENZA PROCESSARE ULTERIORMENTE IL COMANDO
10000 PRINT "      ";LC;">";
10010 ZL=70:GOSUB 60000:PRINT
10020 REM SE LA STRINGA E' VUOTA TORNO AL MODO COMANDO
10030 IF IN$="" THEN RETURN
10040 REM INSERISCO LA STRINGA SULLA LINEA LC, SPOSTANDO LE LINEE DA
10050 REM LC A FT-1 DI UN POSTO VERSO IL BASSO
10060 REM CONTROLLO CHE CI SIA POSTO PER UNA INSERZIONE
10070 IF FT<=DW THEN GOTO 10140
10090 REM NON C'E' SPAZIO PER UNA INSERZIONE
10100 PRINT "▓ATTENZIONE, NON C'E' SPAZIO NELL'AREA"
10110 PRINT "▓▓DI LAVORO. IORNO AL MODO COMANDO."
10120 RETURN
10130 REM POSSO INSERIRE LA LINEA
10140 IF FT=LC THEN GOTO 10180:REM L'AREA DI LAVORO E' VUOTA
10150 FOR I=FT TO LC+1 STEP -1
10160 ED$(I)=ED$(I-1)
10170 NEXT I
```

senza scrivere nulla; in tal caso il programma torna ad attendere un comando (linea 10030). Sempre all'interno di questa routine si controlla se l'area di memoria riservata è completamente piena (linea 10070) ed eventualmente si stampa un avvertimento (linee 10100÷10110); altrimenti si verifica se la linea corrente coincide con la prima linea libera (linea 10140). Se l'esito di quest'ultima verifica è negativo significa che si vuole effettuare un inserimento tra due linee di testo già scritte, ed è quindi necessario spostare tutte le linee successive di una posizione (linee 10150÷10170); viceversa la nuova linea deve essere collocata in coda al testo già scritto, e l'inserzione può essere effettuata direttamente (linea 10180). Sono quindi incrementate le variabili LC e FT prima di tornare ad attendere l'immissione di un'altra linea (linea 10220).

La cancellazione di una o più linee di testo è gestita dalla routine 11000, che inizia passando il controllo alla routine 40000 per determinare quante e quali linee si devono eliminare. Se si è inserito il solo comando «k» sono poste uguali al numero della linea corrente le variabili IN e FI (prima e ultima linea su cui operare), per poter così eliminare questa sola linea. Se il comando «k» è seguito dal simbolo «—» sono eliminate le linee comprese tra quella corrente e l'ultima del testo. Se il comando è seguito da un numero si elimina la linea individuata da quel numero. Se infine il comando è seguito da

```
10180 ED$(LC)=IN$
10190 FT=FT+1
10200 LC=LC+1
10210 REM LEGGO LA PROSSIMA LINEA
10220 GOTO 10000
10990 REM ROUTINE:EFFETTUA LA CANCELLAZIONE DI LINEE DI EDITOR
11000 GOSUB 40000:REM CALCOLO IL RANGE DELLE LINEE DA ELIMINARE
11010 IF IN<=FI THEN GOTO 11070
11020 REM C'E' UN ERRORE NEL COMANDO
11030 PRINT "♠TTENZIONE, RANGE NON BEN DEFINITO"
11040 REM TORNO AL MODO COMANDO
11050 RETURN
11060 REM CONTROLLO CHE I NUMERI SIANO CORRETTI
11070 IF IN>DW OR FI>DW THEN GOTO 11030
11071 IF IN>=FT THEN RETURN:REM NON DEVO ELIMINARE LINEE
11080 REM I NUMERI SONO CORRETTI, EFFETTUO L'ELIMINAZIONE
11090 IF FI>=FT THEN FT=IN:RETURN:REM HO ELIMINATO LE LINEE
11100 FOR I=FI+1 TO FT
11110 ED$(IN+I-FI-1)=ED$(I)
11120 NEXT I
11130 REM ORA METTO A POSTO LA NUOVA FINE TESTO
11140 FT=FT-(FI-IN+1)
11150 IF LC>FT THEN LC=FT
11160 RETURN:TORNO AL MODO COMANDO
11990 REM ROUTINE:LISTA IL CONTENUTO DELL'EDITOR
12000 GOSUB 40000:REM LEGGO RANGE DA LISTARE
12010 REM CONTROLLO LIMITI DEL RANGE
12020 IF IN>=FT THEN RETURN:REM NON C'E' NIENTE DA LISTARE
12030 IF FI>=FT THEN FI=FT-1
12040 IF FI>=IN THEN GOTO 12090
12050 REM AVVERTO DELL'ERRORE
12060 PRINT "♠TTENZIONE, RANGE NON BEN DEFINITO"
12070 RETURN:REM TORNO AL MODO COMANDO
12080 REM LISTO
12090 FOR I=IN TO FI
12100 I$=STR$(I)
12110 REM I$ DEVE ESSERE LUNGO 4 CIFRE
12120 IF LEN(I$)<4 THEN I$=" "+I$:GOTO 12120
12130 PRINT I$;"> ";ED$(I)
12140 NEXT I
12150 REM HO TERMINATO
12160 RETURN
12990 REM ROUTINE:ESEGUE IL COMANDO QUIT
13000 PRINT "XUOI VERAMENTE TERMINARE? [S/N] ";
13010 ZL=1:GOSUB 60000:PRINT
13020 IF IN$<>"S" THEN RETURN
13030 POKE 657,0:REM ABILITO IL TASTO SHIFT/COMMODORE
13040 PRINT "⌉⊠⊠"
13050 PRINT CHR$(142):REM CARATTERI MAIUSCOLI
13052 POKE 53280,14:PRINT "⌈⌉":REM COLORI NORMALI
13060 END
13990 REM ROUTINE:ESEGUE L'EXIT
14000 PRINT "¬AMMI IL NOME DEL FILE SU CUI"
14010 PRINT "⊠MEMORIZZARE L'EDITOR."
14020 PRINT "⊠/OME FILE ? ";
14030 ZL=10:GOSUB 60000:PRINT
14040 IF IN$="" THEN RETURN:REM NON VUOLE TERMINARE
14050 REM MEMORIZZO SU FILE
14060 OPEN 4,8,4,"@0:"+IN$+",S,W"
14070 REM CONTROLLO SE CI SONO LINEE NELL'EDITOR
14080 IF FT=0 THEN GOTO 14130
14090 REM MEMORIZZO
14091 LL=LEN(ED$(I))
14092 FOR I=0 TO FT-1
14093 LL=LEN(ED$(I))
14101 FOR J=1 TO LL
14102 JJ=J-1
14103 LJ=LL-J
14104 REM SOSTITUZIONE DI EVENTUALI CODICI DI CONTROLLO DRIVER
14105 IF MID$(ED$(I),J,1)<>","AND MID$(ED$(I),J,1)<>":" THEN 14108
14106 GOSUB 63010
14108 NEXT J
14110 PRINT#4,ED$(I)
14120 NEXT I
14130 CLOSE 4:REM CHIUDO IL CANALE
14140 REM ORA SALTO ALLA ROUTINE DI QUIT
14150 GOSUB 13000
14160 RETURN
14990 REM ROUTINE:IN ALTO DI N LINEE
15000 N=VAL(RIGHT$(IN$,LEN(IN$)-1)):REM HO LETTO IL NUMERO CHE SEGUE LA U
15010 REM SE IL NUMERO E' ZERO,VADO SU DI 1 SOLA LINEA
15020 IF N=0 THEN N=1
15030 REM DECREMENTO IL PUNTATORE ALLA LINEA CORRENTE
15040 LC=LC-N
15050 REM CONTROLLO CHE LC SIA >= 0 E <=FT
15060 IF LC<0 THEN LC=0
15061 IF LC>FT THEN LC=FT
15070 REM STAMPO LA LINEA USANDO UNA PARTE DELLA ROUTINE CHE LISTA LE LINEE
15080 IN=LC
15090 FI=LC
```

due numeri separati dal simbolo «—» si eliminano le linee comprese tra i due valori forniti. Il controllo torna poi alla linea 11010 che verifica se sono stati immessi valori non corretti; in questa eventualità è visualizzato un messaggio di avvertimento (linea 11030) prima di tornare ad attendere un comando (linea 11050). Si controlla poi se la prima linea da eliminare è effettivamente scritta (linea 11071) e si effettua la cancellazione (linee 11100÷11120). Prima di tornare ad attendere un comando (linea 11160) si determina il nuovo valore di FT ed eventualmente quello di LC (linee 11140÷11150).
La routine 12000, che visualizza una o più linee di testo, inizia chiamando la routine 40000 per poter determinare quante e quali linee devono essere visualizzate. Controlla poi se sono stati immessi valori non corretti (linee 12040÷12070) e stampa sul video le linee richieste (linee 12090÷12140).
La routine 13000 gestisce la fine della sessione di lavoro. Dopo aver chiesto se si ha veramente intenzione di terminare (linea 13000), questa routine riabilita il cambio dei caratteri (linea 13040) e li pone in maiuscolo prima di concludere l'esecuzione del programma.
Se si sceglie di memorizzare su disco il testo prima di finire la sessione di lavoro, il programma chiama la routine 14000 che, chiesto il nome che si vuole dare al testo appena composto (linee 14000÷14020), apre un file su disco (linea 14060) e vi memorizza il

```
15100 GOTO 12020
15990 REM ROUTINE:IN BASSO DI N LINEE
16000 N=VAL(RIGHT$(IN$,LEN(IN$)-1)):REM HO LETTO IL NUMERO CHE SEGUE LA D
16010 REM SE IL NUMERO E' ZERO,VADO GIU' DI 1 SOLA LINEA
16020 IF N=0 THEN N=1
16030 REM INCREMENTO IL PUNTATORE ALLA LILINEA CORRENTE
16040 LC=LC+N
16050 REM CONTROLLO CHE LC SIA <= FT E SIA >=0
16060 IF LC>FT THEN LC=FT
16061 IF LC<0 THEN LC=0
16070 REM STAMPO LA LINEA USANDO UNA PARTE DELLA ROUTINE CHE LISTA LE LINEE
16080 IN=LC
16090 FI=LC
16100 GOTO 12020
16990 REM ROUTINE:LISTA SU STAMPANTE
17000 GOSUB 40000:REM LEGGO RANGE DA LISTARE
17010 REM CONTROLLO LIMITI DEL RANGE
17020 IF IN>=FT THEN RETURN:REM NON C'E' NIENTE DA LISTARE
17030 IF FI>=FT THEN FI=FT-1
17040 IF FI>=IN THEN GOTO 17090
17050 REM AVVERTO DELL'ERRORE
17060 PRINT "♠TTENZIONE, RANGE NON BEN DEFINITO"
17070 RETURN
17080 REM LISTO SU STAMPANTE
17090 OPEN4,4,7
17100 FOR I=IN TO FI
17110 PRINT#4,ED$(I)
17120 NEXT I
17130 REM HO TERMINATO
17140 CLOSE 4
17150 RETURN
17990 REM ROUTINE:CAMBIA IL NUMERO DI LINEA CORRENTE
18000 LC=VAL(RIGHT$(IN$,LEN(IN$)-1))
18010 REM CONTROLLO CHE NON FUORIESCA DAI LIMITI
18020 IF LC>FT THEN LC=FT
18030 IF LC<0 THEN LC=0
18040 REM HO TERMINATO
18050 RETURN
18990 REM ROUTINE:LEGGE UN FILE DA DISCO
19000 PRINT "/OME DEL FILE DI INGRESSO ? ";
19010 ZL=10:GOSUB 60000:PRINT
19020 IF IN$="" THEN RETURN:REM NESSUN FILE
19030 REM APRO IL CANALE DEI COMANDI
19040 OPEN15,8,15
19050 REM APRO IL CANALE PER IL FILE
19060 OPEN 4,8,4,IN$+",S,R"
19070 REM LEGGO SE CI SONO ERRORI
19080 INPUT#15,E,E$,T,S
19090 IF E<20 THEN GOTO 19200
19100 REM CI SONO ERRORI
19110 PRINT "‾RRORE NELL'APERTURA DEL FILE"
19120 PRINT "/UMERO     :";E
19130 PRINT "¬ODICE     :";E$
19140 PRINT "¦RACCIA    :";T
19150 PRINT "♠ETTORE    :";S
19160 REM RIINIZIALIZZO IL DISCO
19170 PRINT#15,"I"
19171 CLOSE 4:CLOSE 15
19180 GOTO 19000:REM RICHIEDO IL NOME DEL FILE
19190 REM NON CI SONO ERRORI, LEGGO IL FILE
19200 FT=0
19210 INPUT#4,ED$(FT)
19211 LL=LEN(ED$(FT))
19212 FOR J=1 TO LL
19213 JJ=J-1:LJ=LL-J
19215 IF MID$(ED$(FT),J,1)<>"]"AND MID$(ED$(FT),J,1)<>"[" THEN 19218
19216 REM RICONVERSIONE EVENTUALI CARATTERI SOSTITUITI
19217 GOSUB 63060
19218 NEXT J
19220 IF ST<>0 THEN CLOSE 4:CLOSE 15:FT=FT+1:RETURN:REM HO LETTO IL FILE
19230 FT=FT+1
19240 GOTO 19210
19990 REM ROUTINE:CERCA UNA STRINGA DALLA POSIZIONE LC ALLA FINE DEL TESTO
20000 PRINT "▒♠TRINGA DA CERCARE ? ";
20010 ZL=70:GOSUB 60000:PRINT
20020 IF IN$="" THEN RETURN:REM TORNO SENZA CERCARE
20120 GOSUB 42000:REM CERCO LA STRINGA IN ED$, RISULTATO IN FG
20130 IF FG=1 THEN GOTO 20170
20140 PRINT "▒▒♠TRINGA NON TROVATA"
20150 RETURN
20160 REM STRINGA TROVATA:STAMPO LA LINEA CHE LA CONTIENE
20170 IN=I
20180 FI=I
20190 REM SALTO ALL'INTERNO DELLA ROUTINE CHE LISTA
20200 GOTO 12020
20210 REM LA ROUTINE E' TERMINATA
20990 REM ROUTINE:SOSTITUISCE UNA STRINGA CON UN'ALTRA
21000 PRINT "▒×ECCHIA STRINGA ? ";
21010 ZL=70:GOSUB 60000:PRINT
21020 IF IN$="" THEN RETURN
```

testo (linee 14100÷14130) per poi chiamare la routine 13000.
Il comando «u» è gestito dalla routine 15000 che, letto il numero che segue il comando (linea 15000), assume come nuova linea corrente quella che precede la linea attuale di un numero di righe pari a quello indicato. È poi decrementata la variabile LC in base al numero inserito con il comando (linea 15040); si controlla che il nuovo valore di LC non individui una linea al di fuori del testo (linee 15060, 15061) e si visualizza la nuova linea corrente utilizzando parte della routine 12000.
Il comando «d» è gestito dalla routine 16000, che effettua gli stessi passi della 15000, con la sola differenza che il valore di LC è incrementato (linea 16040) anziché decrementato, in quanto si deve spostare la linea corrente di una o più posizioni verso la fine del testo (anziché verso l'inizio).
La routine 17000, che permette di stampare una o più linee di testo su carta, ha la medesima struttura della 12000. Le uniche istruzioni diverse sono ovviamente quelle che aprono e chiudono i canali di comunicazione con le periferiche (la stampante invece del disco). Per cambiare la linea corrente la routine 18000 deve solo controllare che il numero di linea fornito insieme al comando «c» non sia al di fuori del testo e determinare il nuovo valore di LC in base al numero introdotto.
La routine 19000, che carica un file da disco, chiede il nome del file, apre il canale

```
21030 REM LA CERCO
21040 GOSUB 42000
21050 IF FG=0 THEN PRINT "/ON ESISTE":RETURN
21060 REM E' STATA TROVATA, LA DEVO SOSTITUIRE
21070 BU$=IN$
21080 PRINT "▒▌ROVATA"
21081 PRINT STR$(I);"> ";ED$(I)
21090 PRINT "▒/UOVA STRINGA ? ";
21100 ZL=70:GOSUB 60000:PRINT
21110 REM SOSTITUISCO
21120 REM CONTROLLO CHE LA LUNGHEZZA TOTALE DELLA NUOVA STRINGA
21130 REM NON SUPERI I 255 CARATTERI
21140 IF LEN(ED$(I))-LEN(BU$)+LEN(IN$)<=255 THEN GOTO 21200
21150 REM NON POSSO EFFETTUARE L'INSERZIONE
21160 PRINT "▒♥OSTITUZIONE IMPOSSIBILE."
21170 PRINT "▒♥TRINGA TROPPO LUNGA."
21180 RETURN
21190 REM EFFETTUO LA SOSTITUZIONE
21200 ED$(I)=LEFT$(ED$(I),J-1)+IN$+RIGHT$(ED$(I),LEN(ED$(I))-LEN(BU$)-J+1)
21210 REM STAMPO LA NUOVA LINEA
21220 IN=I
21230 FI=I
21240 REM VADO NELLA ROUTINE LIST
21250 GOTO 12020
21990 REM ROUTINE:VISUALIZZA LA MASCHERA DI HELP
22000 PRINT "♫";TAB(10);"¬OMANDI DELL'EDITOR"
22010 PRINT:PRINT
22020 PRINT "I:";TAB(10);"╲NSERISCE UN TESTO"
22030 PRINT "K:";TAB(10);"¬ANCELLA LINEE DI TESTO"
22040 PRINT "L:";TAB(10);"LISTA IL TESTO"
22050 PRINT "Q:";TAB(10);"|ERMINA LA SESSIONE"
22060 PRINT "E:";TAB(10);"|ERMINA E SALVA IL TESTO"
22070 PRINT "U:";TAB(10);"×A VERSO L'ALTO"
22080 PRINT "D:";TAB(10);"×A VERSO IL BASSO"
22090 PRINT "P:";TAB(10);"LISTA SU STAMPANTE"
22100 PRINT "C:";TAB(10);"¬AMBIA LA LINEA CORRENTE"
22110 PRINT "M:";TAB(10);"¬ARICA UN TESTO DA DISCO"
22120 PRINT "F:";TAB(10);"¬ERCA UNA STRINGA"
22130 PRINT "S:";TAB(10);"♥OSTITUISCE UNA STRINGA"
22140 PRINT "H:";TAB(10);"×ISUALIZZA QUESTA MASCHERA"
22150 POKE 214,20:PRINT
22160 PRINT "▒L¬| ⁄—/▒ PER PROSEGUIRE"
22170 GET IN$:IF IN$<>CHR$(13) THEN GOTO 22170
22180 PRINT "♫";
22190 RETURN
39980 REM ROUTINE:LEGGE LA PARTE DESTRA DELLA STRINGA DI COMANDO CHE CONTIENE
39990 REM IL RANGE DELLE LINEE DA PROCESSARE
40000 IN$=RIGHT$(IN$,LEN(IN$)-1)
40010 IF IN$="" THEN IN=LC:FI=LC:RETURN:REM PUNTO LA LINEA CORRENTE
40020 REM PROCESSO IN$ PER VEDERE IL RANGE DELLE LINEE
40030 REM C'E' LA PRIMA CIFRA?
40040 CI$=""
40050 I=1
40060 TM$=MID$(IN$,I,1)
40070 IF NOT(TM$>="0" AND TM$<="9") THEN GOTO 40110
40080 CI$=CI$+TM$
40090 I=I+1:IF I<=LEN(IN$) THEN GOTO 40060
40100 REM IN CI$ HO UNA CIFRA OPPURE HO IL CARATTERE NULL(ZERO)
40110 IN=VAL(CI$)
40120 REM ORA CONTROLLO SE C'E' LA SECONDA CIFRA
40121 IF I>LEN(IN$) THEN FI=IN:RETURN:REM C'E' SOLO UNA CIFRA SENZA SEPARATORE
40130 IN$=RIGHT$(IN$,LEN(IN$)-I)
40140 REM HO TOLTO LA PRIMA CIFRA E L'EVENTUALE SEPARATORE
40150 IF IN$="" THEN FI=FT-1:RETURN:REM MANCA LA SECONDA CIFRA
40160 REM LEGGO LA SECONDA CIFRA
40170 CI$=""
40180 I=1
40190 TM$=MID$(IN$,I,1)
40200 IF NOT(TM$>="0" AND TM$<="9") THEN GOTO 40240
40210 CI$=CI$+TM$
40220 I=I+1:IF I<=LEN(IN$) THEN GOTO 40190
40230 REM HO LETTO LA SECONDA CIFRA
40240 FI=VAL(CI$)
40250 REM ORA HO LE DUE CIFRE IN IN E FI, TERMINA LA PROCEDURA
40260 RETURN
40990 REM ROUTINE:TOGLIE GLI SPAZI BIANCHI NELLA STRINGA IN$
41000 I=1
41010 IF MID$(IN$,I,1)<>" " THEN GOTO 41040
41020 IN$=LEFT$(IN$,I-1)+RIGHT$(IN$,LEN(IN$)-I)
41030 GOTO 41010
41040 I=I+1:IF I<=LEN(IN$) THEN GOTO 41010
41050 REM HO TOLTO GLI SPAZI BIANCHI NELLA STRINGA DI COMANDO
41060 RETURN
41980 REM ROUTINE:CERCA LA STRINGA IN$ NELL'ARRAY ED$ E PONE LA VARIABILE
41990 REM FG A ZERO OD A UNO A SECONDA DEL RISULTATO DELLA RICERCA
42000 FG=0
42010 IF LC=FT THEN RETURN
42020 REM INIZIA LA RICERCA
42030 I=LC
42040 BU$=ED$(I)
```

di comando (linea 19040) e quello del file (linea 19060), controlla se si sono verificati errori (linea 19080) ed eventualmente visualizza il codice dell'errore (linee 19090÷19180). Se non si sono verificati errori carica il file e chiude i canali aperti in precedenza.
La routine 20000 cerca una stringa di caratteri nel testo a partire dalla linea corrente fino alla prima linea vuota. Questa routine inizia chiedendo quale stringa si vuole cercare; se non è specificata alcuna stringa ma si digita solo il tasto RETURN il programma si pone di nuovo in attesa di un comando (linea 12020).
Il controllo passa poi alla routine 42000, che cerca la stringa confrontandola con i caratteri contenuti in tutte le linee del testo interessate, e pone ad uno la variabile booleana FG se trova la stringa specificata. Se la stringa non è stata trovata è visualizzato un messaggio di avvertimento (linea 20140), altrimenti si stampa la linea che la contiene utilizzando parte della routine 12000 (linea 20200).
La sostituzione di una stringa con un'altra è gestita dalla routine 21000 che, chiesta la stringa da sostituire, si avvale della routine 42000 per trovarla (linea 21040). Se la stringa è individuata si chiede di digitare la nuova stringa (linea 21090) e, dopo aver controllato che la linea così composta non superi la lunghezza di 255 caratteri (linea 21140), si effettua la sostituzione (linea 21200) e si visualizza la nuova linea (linea 21250).
La routine 22000, infine, visualizza la spiegazione dei comandi disponibili (linee 22020÷22150).

```
42050 GOSUB 43000:REM CERCO LA STRINGA IN$ IN BU$:RISULTATO IN FG
42060 IF FG=1 THEN RETURN
42070 I=I+1
42080 IF I<FT THEN GOTO 42040
42090 REM LA RICERCA E' TERMINATA
42100 RETURN
42980 REM ROUTINE:CERCA LA STRINGA IN$ NELLA STRINGA BU$ E PONE LA VARIABILE
42990 REM FG A ZERO OD A UNO A SECONDA DEL RISULTATO DELLA RICERCA
43000 FG=0
43010 IF LEN(IN$)>LEN(BU$) THEN RETURN:REM NON PUO' ESSERCI
43020 J=1
43030 IF IN$=MID$(BU$,J,LEN(IN$)) THEN FG=1:RETURN
43040 J=J+1
43050 IF J<=LEN(BU$)-LEN(IN$)+1 THEN GOTO 43030
43060 RETURN:REM NON TROVATA
59920 REM ROUTINE:GESTISCE L'INPUT DI UNA STRINGA ALFANUMERICA
59930 REM LE VARIABILI UTILIZZATE SONO:Z6,Z7,Z8,Z9,Z8$,IN$
59940 REM IN INGRESSO VUOLE IL VALORE ZL CHE E' LA LUNGHEZZA MASSIMA DELLA
59950 REM STRINGA DA LEGGERE
59960 REM IN USCITA DA LA STRINGA LETTA IN IN$ E LA SUA LUNGHEZZA IN Z9
59980 REM CANCELLO LA ZONA DI SCHERMO IN CUI VERRA' DIGITATA LA STRINGA
60000 IN$="":Z7=TI
60040 REM LEGGO UN CARATTERE
60060 GET Z8$:IF Z8$<>"" THEN 60160
60080 REM ACCENSIONE E SPEGNIMENTO DEL CURSORE
60100 IF Z7<TI AND NOT(Z6) THEN PRINT "▒▌";:Z6=NOT(Z6):Z7=TI+15
60110 IF Z7<TI AND Z6 THEN PRINT " ▌";:Z6=NOT(Z6):Z7=TI+15
60120 GOTO 60060
60140 REM E' STATO DIGITATO UN CARATTERE
60160 Z8=ASC(Z8$):Z9=LEN(IN$)
60165 IF Z8=34 THEN PRINT Z8$;CHR$(20);
60180 REM SE NON E' UN CARATTERE ALFANUMERICO, DEVE ESSERE UN RETURN O DELETE
60200 IF NOT((Z8>31 AND Z8<96) OR (Z8>192 AND Z8<219)) THEN GOTO 60320
60220 REM CONTROLLO CHE NON SIA STATA SUPERATA LA LUNGHEZZA MASSIMA
60240 IF Z9=ZL THEN GOTO 60060
60260 REM LO AGGIUNGO ALLA STRINGA IN$
60280 IN$=IN$+Z8$:PRINT Z8$;:GOTO 60060
60300 REM SE E' UN RETURN, HO TERMINATO LA LETTURA
60320 IF Z8=13 THEN PRINT " ▌";:RETURN
60340 REM SE E' DELETE, CANCELLO IN IN$ E SUL VIDEO L' ULTIMO CARATTERE DIGITATO
60360 IF Z8=20 AND Z9>0 THEN IN$=LEFT$(IN$,Z9-1):PRINT " ▌▌";:GOTO 60060
60370 GOTO 60060
60960 REM ROUTINE:INIZIALIZZAZIONE COSTANTI
60980 REM CIRCUITO VIDEO
61000 VI=53248
61020 REM CIRCUITO SUONO
61040 SI=54272
61060 REM MEMORIA VIDEO
61080 MV=1024
61100 REM MEMORIA COLORE
61120 MC=55296
61140 REM COSTANTI DI USO COMUNE
61160 ZL=9
61180 REM INIZIALIZZAZIONE CHIP SUONO
61200 FOR I=0 TO 24
61210 POKE SI+I,0
61220 NEXT I
61230 RETURN
61980 REM ROUTINE:STAMPA IL TITOLO DEL PROGRAMMA MEMORIZZATO IN PG$
62000 GOSUB 61000
62010 POKE VI+32,15
62020 POKE VI+33,15
62030 PRINT "▒▒▒▒▒▒";
62040 PRINT TAB(6);"┌──────────────────────────┐"
62050 FOR I=1 TO 5
62060 PRINT TAB(6);"│                          │"
62070 NEXT I
62080 PRINT TAB(6);"└──────────────────────────┘"
62090 PRINT TAB(6);"▒▒▒▒▒▒▒▒DIGITA ▒RETURN▒ PER PROSEGUIRE"
62100 PRINT "▒▒▒▒▒▒"
62110 FOR I=1 TO 5
62120 PRINT TAB(7);
62130 FOR J=1 TO 26
62140 PRINT "▒▒ ";
62150 NEXT J
62160 PRINT
62170 NEXT I
62190 REM ORA SCRIVO IL TITOLO
62210 PRINT "▒▒▒▒▒▒▒▒▒";TAB((40-LEN(PG$))/2);"▒▒";PG$
62220 GET Z9$
62230 IF Z9$<>CHR$(13) THEN GOTO 62220
62240 PRINT "▒▒";
62250 RETURN
63000 REM ROUTINE: SOSTITUZIONE DI EVENTUALI CODICI DI CONTROLLO DRIVER
63010 IF MID$(ED$(I),J,1)=","THEN EE$=LEFT$(ED$(I),JJ)+"]"+RIGHT$(ED$(I),LJ)
63020 IF MID$(ED$(I),J,1)=":"THEN EE$=LEFT$(ED$(I),JJ)+"["+RIGHT$(ED$(I),LJ)
63030 ED$(I)=EE$:RETURN
63050 REM ROUTINE: RICONVERSIONE EVENTUALI CARATTERI SOSTITUITI
63060 IF MID$(ED$(FT),J,1)="]"THEN EE$=LEFT$(ED$(FT),JJ)+","+RIGHT$(ED$(FT),LJ)
63070 IF MID$(ED$(FT),J,1)="["THEN EE$=LEFT$(ED$(FT),JJ)+":"+RIGHT$(ED$(FT),LJ)
63080 ED$(FT)=EE$:RETURN
```

Le otto regine

Con questo programma presenteremo un solitario che richiederà riflessione ed applicazione. Il gioco consiste nel posizionare otto regine su una scacchiera in modo tale che nessuna di esse sia minacciata da un'altra. Chiunque conosca il gioco degli scacchi sa che la regina si può muovere in orizzontale, in verticale e in diagonale: poiché una scacchiera è composta da 8×8=64 caselle potrebbe sembrare impossibile posizionare tutte le otto regine in modo che non ve ne siano due sotto scacco reciproco. In realtà esistono circa settanta soluzioni diverse al problema (senza contare le simmetrie e le rotazioni), ma il giocatore si accorgerà che non è poi così facile trovarne una.

Analisi del problema

Per realizzare questo gioco si utilizzeranno tutti gli otto sprites disponibili nel CBM-64 per rappresentare le regine. Sarà così presentato un esempio di come si possa gestire il movimento di uno sprite sullo schermo, entrando nella problematica dell'animazione delle immagini. In questo caso lo sprite sarà semplicemente spostato sullo schermo, ma il lettore potrà divertirsi nel provare ad ottenere un'animazione più sofisticata definendo più sprites da sostituire l'uno con l'altro in rapida successione.

Il programma dovrà presentare sul video una scacchiera di 8 colonne per 8 righe e, al suo fianco, gli otto sprites che raffigurano le regine e la descrizione dei comandi disponibili. Il giocatore potrà selezionare la regina che desidera posizionare sulla scacchiera, muoverla in orizzontale e verticale, riportarla al di fuori della scacchiera e, decisa la casella dove la vuole mettere, immobilizzarla per selezionare poi un'altra regina. Non sarà però possibile scegliere una nuova regina fino a che non si è posizionata quella in gioco su una casella che non sia sotto lo scacco di una regina già inserita. Sarà invece possibile scegliere una regina già posizionata e spostarla in un'altra casella della scacchiera o ricondurla alla posizione di partenza.

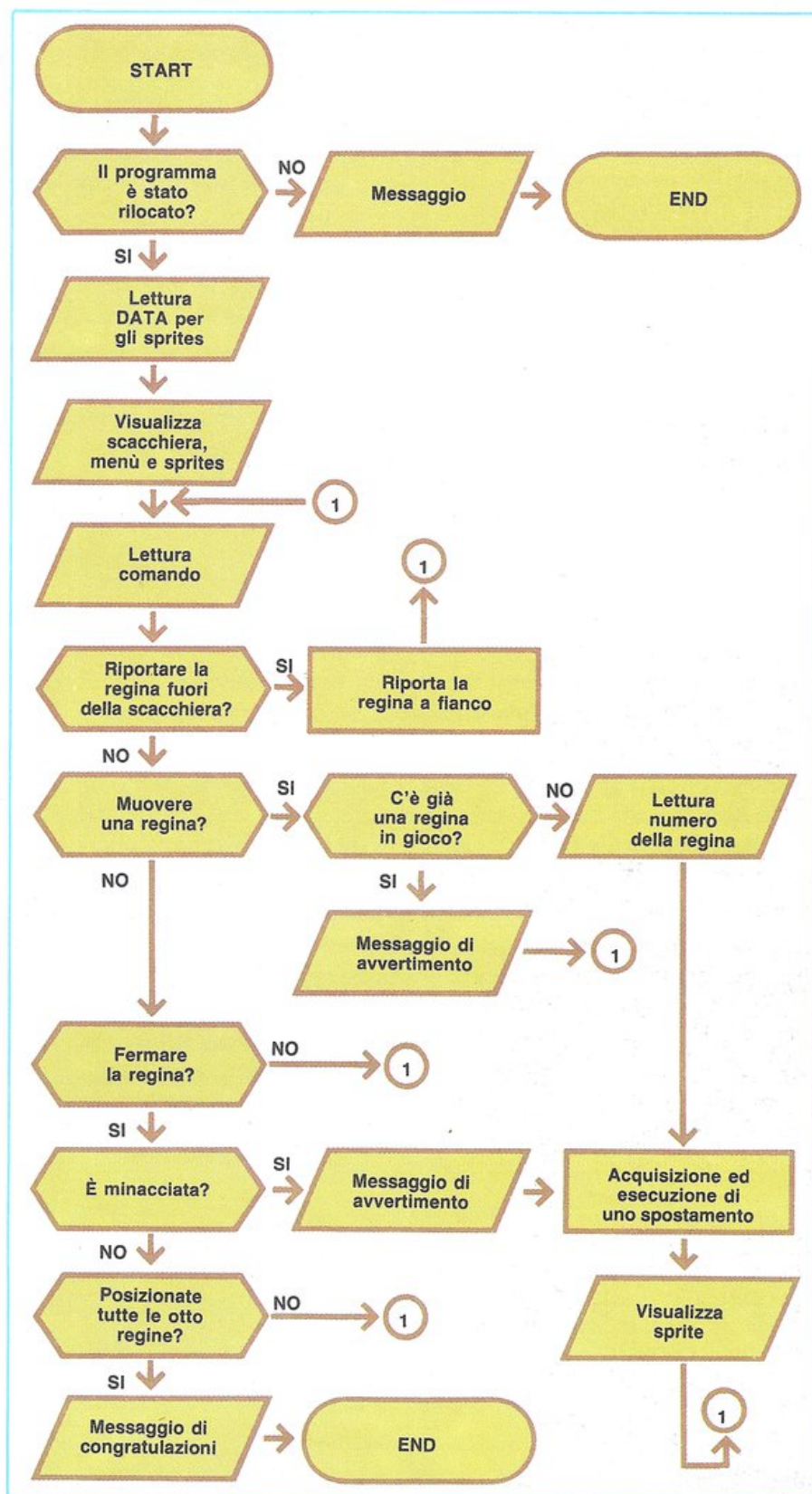
Per rappresentare la scacchiera sarà utilizzata la possibilità di inserire i tasti funzione nelle istruzioni PRINT dei programmi.

Le regine, invece, saranno raffigurate, come detto, mediante gli sprites; questi hanno l'utile caratteristica di potersi sovrapporre a qualunque carattere già esistente muovendosi sul video senza interferire con quanto vi è contenuto. Dopo aver letto il disegno dello sprite dalle istruzioni DATA, il programma dovrà comunicare, con istruzioni POKE, alla scheda d'interfaccia video (che traduce i segnali del computer in segnali che producono un'immagine sul televisore) le coordinate del punto dove posizionare lo sprite. Per muovere la regina sarà poi sufficiente fornire la nuova posizione sempre con istruzioni POKE.

Dopo che il giocatore avrà scelto la casella in cui posizionare la regina, il programma dovrà controllare se questa casella si trova sotto scacco. Per controllare uno scacco orizzontale o verticale basterà verificare se esiste un'altra regina che ha la stessa coordinata Y o X di quella che si vuole posizionare (la coordinata Y corrisponde al numero di riga mentre la coordinata X corrisponde al numero di colonna). Per verificare invece un eventuale scacco diagonale il programma dovrà effettuare le seguenti operazioni: sottrarre la coordinata X di un'altra regina a quella della regina da posizionare, sottrarre la coordinata Y dell'altra regina a quella da posizionare, ricavare i valori assoluti dei risultati di queste due sottrazioni e verificare se sono uguali tra di loro; se i due valori assoluti coincidono allora le due regine si trovano sulla stessa diagonale e si è quindi verificato uno scacco.

Diagrammi di flusso

Il programma inizia controllando che la sua locazione di base sia stata spostata. Quindi si passa alla lettura dei DATA che definiscono gli sprites. Il passo successivo è la visualizzazione della scacchiera, delle otto regine e dei comandi permessi. Quindi il programma legge il comando digitato. Nel seguito si esamina se il comando impostato riguarda la ricollocazione di una regina al suo posto di partenza. Questo comando è poi eseguito senza ulteriori verifiche. Se viceversa il giocatore intende muovere una regina, il comando (e le operazioni che ne conseguono) è eseguito solo se l'eventuale regina mossa precedentemente è stata fermata. Nel blocco successivo il programma controlla se il giocatore intende fermare la regina e, in caso positivo, verifica se la posizione in cui la vuole fermare è minacciata. Qualora si abbia una collisione (posizione sotto scacco) il programma non accetta il fermo e richiede un ulteriore spostamento. Nel test successivo si verifica se tutte le regine sono state collocate sulla scacchiera. In caso positivo il giocatore ha finito il solitario, e appare un messaggio di congratulazioni; viceversa il programma torna ad accettare un nuovo comando.

Il programma

Dopo la presentazione del titolo e l'inizializzazione delle costanti del C-64, si comunica all'utente che è necessario spostare la base del programma prima del caricamento in memoria (linee 170÷280). Questa operazione è necessaria per rendere disponibile un'area in cui memorizzare le figure delle otto regine leggendo il loro disegno dalle istruzioni DATA (linee 310÷340). Sono poi inizializzati gli otto sprites raffiguranti le regine (linee 360÷430), è assegnato un diverso colore per ciascuno di essi (linee 440, 450) e sono determinati i puntatori che serviranno ad inviduarli (linee 462÷464). Alle linee 510÷580 si trova il doppio ciclo che permette la visualizzazione della scacchiera; le caselle sono stampate alternativamente in nero e bianco, utilizzando le funzioni che selezionano questi due colori e gli indici dei cicli; in questo modo dopo una casella nera se ne trova una bianca, poi un'altra nera, ecc. Si dimensiona poi il vettore SP, che conterrà le due coordinate (espresse in punti video) di ciascuna delle otto regine (linea 610), e sono inizializzate le posizioni di partenza degli sprites (linee 630÷700). La linea 712 chiama la routine 9000, che posiziona gli sprites utilizzando le coordinate memorizzate nel vettore SP; la linea 770 accende quindi gli sprites. È poi visualizzata la

```
0 REM ****************
1 REM *LE OTTO REGINE*
2 REM ****************
4 REM VARIABILI UTILIZZATE
6 REM PG$       :TITOLO DEL PROGRAMMA
7 REM I,J       :CONTATORI DI CICLO
8 REM A         :CONTIENE IL CODICE DEL TASTO DIGITATO
9 REM QU$       :SE STAMPATO, DISEGNA UN QUADRATO 3*3
10 REM CO$      :COLORE DEL QUADRATO DA DISEGNARE
11 REM R        :NUMERO DELLA REGINA CHE SI STA MUOVENDO
12 REM FI       :FLAG, SE 1 INDICA LA FINE DELLA FASE DI SPOSTAMENTO REGINA
13 REM CO       :FLAG, SE 1 INDICA CHE CI SONO COLLISIONI SULLA SCACCHIERA
14 REM IN$      :CONTIENE LA STRINGA DIGITATA
15 REM X,Y      :COORDINATE FISICHE DELLA REGINA R-ESIMA
16 REM X1,Y1    :COORDINATE FISICHE DI TUTTE LE ALTRE REGINE, UNA PER VOLTA
17 REM SP(8,2)  :MATRICE CHE MEMORIZZA LE COORDINATE DELLE 8 REGINE
100 REM TITOLO ED INIZIALIZZAZIONE COSTANTI C64
110 PG$="LE OTTO REGINE"
120 GOSUB 62000
130 REM CONTROLLO CHE LA BASE DEL PROGRAMMA BASIC SIA ALLA LOCAZIONE 16384
140 IF PEEK(44)=64 THEN GOTO 301
150 REM AVVERTO CHE IL PROGRAMMA NON PUO' ANDARE IN ESECUZIONE
160 PRINT "■"
170 PRINT "ATTENZIONE, HAI DIMENTICATO"
180 PRINT "■■DI IMMETTERE I SEGUENTI COMANDI:"
190 PRINT "■■■1) POKE 44,64"
200 PRINT "■■2) POKE 16384,0"
210 PRINT "■■PRIMA DI IMMETTERLI RICORDA"
220 PRINT "■DI SALVARE IL PROGRAMMA, SE QUESTO"
230 PRINT "■NON E' GIA' PRESENTE SU NASTRO,"
240 PRINT "■E POI DI RICARICARLO"
280 END
290 REM INIZIALIZZAZIONE E MASCHERA VIDEO
300 REM CARICO IN MEMORIA LE FIGURE DELLE 8 REGINE
301 PRINT "ATTENDERE UN ATTIMO, PER FAVORE"
310 FOR I=0 TO 64*8-1
320 READ A
330 POKE 2048+I,A
340 NEXT I
350 REM INIZIALIZZO GLI 8 SPRITES
360 FOR I=0 TO 16
370 POKE VI+I,0
380 NEXT I:REM LOCAZIONI 0,0
390 POKE VI+21,0:REM TUTTI SPENTI
400 POKE VI+23,0:REM NESSUNO ESPANSO IN VERTICALE
410 POKE VI+29,0:REM NESSUNO ESPANSO IN ORIZZONTALE
420 POKE VI+27,0:REM PRIORITA' AGLI SPRITE SULLO SFONDO
430 POKE VI+28,0:REM SPRITE MONOCOLORI
440 FOR I=53287 TO 53294
450 POKE I,I-53285
460 NEXT I:REM HO ASSEGNATO I COLORI AGLI SPRITES
461 REM INIZIALIZZO I PUNTATORI AGLI SPRITES
462 FOR I=0 TO 7
463 POKE 2040+I,32+I
464 NEXT I
469 REM DISEGNO LA SCACCHIERA
470 PRINT "■":REM CARATTERI BIANCHI
480 PRINT "■";
500 QU$="■   ■■■■   ■■■■   ■■":REM QUADRATO 3*3
510 FOR I=1 TO 8
515 FOR J=1 TO 8
520 REM DEVO DISEGNARE 8 QUADRATI BIANCHI E NERI ALTERNATIVAMENTE
530 CO$="■"
540 IF (J+I)-INT((J+I)/2)*2=0 THEN CO$="■"
550 PRINT CO$;QU$;
560 NEXT J
570 PRINT "■■"
580 NEXT I
590 REM HO DISEGNATO LA SCACCHIERA
600 REM DIMENSIONO ARRAY
610 DIM SP(8,2)
620 REM INIZIALIZZO LE POSIZIONI DEGLI SPRITES
630 SP(1,1)=220:SP(1,2)=52
640 SP(2,1)=220:SP(2,2)=76
650 SP(3,1)=220:SP(3,2)=100
660 SP(4,1)=220:SP(4,2)=124
670 SP(5,1)=220:SP(5,2)=148
680 SP(6,1)=220:SP(6,2)=172
690 SP(7,1)=220:SP(7,2)=196
700 SP(8,1)=220:SP(8,2)=220
710 REM POSIZIONO GLI SPRITES
711 FOR R=1 TO 8
712 GOSUB 9000
713 NEXT R
760 REM ACCENDO GLI SPRITES
770 POKE VI+21,255
780 REM GLI SPRITE POSSONO STARE DALLA POSIZIONE 28 ALLA 196 IN ORIZZONTALE
790 REM E DALLA 52 ALLA 220 IN VERTICALE
800 REM ORA INIZIA IL PROGRAMMA
810 REM VISUALIZZO LA ZONA DI AIUTO
```

spiegazione dei comandi disponibili (linee 830÷930), sono inizializzate le variabili di controllo (linea 940) ed è atteso un comando (linea 960). Se l'utente chiede di selezionare una regina (linea 970), il controllo passa alla routine 8000, che verifica innanzitutto se c'è un'altra regina già in gioco ma non ancora posizionata (linea 8000), nel qual caso è visualizzato un messaggio (linea 8010) e si attende un altro comando. Altrimenti si chiede quale regina si intende selezionare (linee 8060, 8070) e si controlla che il valore fornito corrisponda ad una delle otto figure disponibili (linea 8110).
Il programma visualizza poi il numero della regina scelta (linea 8180), pone a zero la variabile di controllo FI, che serve ad impedire di selezionare più di una regina alla volta (linea 8181), e torna ad attendere un comando.
La possibilità di riportare una figura nella posizione di partenza è gestita dalla routine 11000 che, determinate le coordinate della casella in cui deve essere riposizionata la regina in base al suo numero (linee 11000, 11010), esegue il comando (linee 11020, 11030). Sono poi poste a zero le variabili R (numero della regina selezionata) e CO. La prima serve anche ad abilitare i tasti che permettono lo spostamento nella scacchiera, e la seconda si usa per controllare se si è verificata una collisione, cioè uno scacco. Inoltre è posta ad uno la variabile FI (linea 11050).
Qualora invece sia stato premuto il tasto F per

```
820 POKE 214,1:PRINT
830 PRINT SPC(28);"P      ALTO"
840 PRINT SPC(28);".     BASSO"
850 PRINT SPC(28);";    DESTRA"
860 PRINT SPC(28);"L  SINISTRA"
870 PRINT "▓▓";SPC(28);"M  MUOVI"
880 PRINT SPC(31);"REGINA"
890 PRINT "▓";SPC(28);"F  FERMA"
900 PRINT SPC(31);"REGINA"
910 PRINT "▓";SPC(28);"E  FUORI"
920 PRINT SPC(31);"REGINA"
922 PRINT "▓";SPC(28);"T  TERMINA"
924 PRINT SPC(31);"IL GIOCO"
930 REM LA REGINA DA MUOVERE PER ORA NON ESISTE
940 R=0:FI=1:CO=0
950 REM LEGGO UN COMANDO
960 GET IN$
965 IF IN$="T" THEN GOSUB 13130
970 IF IN$="M" THEN GOSUB 8000:IN$="":GOTO 1010
980 IF IN$="E" THEN GOSUB 11000:IN$="":GOTO 1010
990 IF IN$="F" THEN GOSUB 12000:IN$="":IF CO=0 THEN GOSUB 13000
1000 REM SE C'E' UNA REGINA DA SPOSTARE, LA SPOSTO
1010 IF R<>0 THEN GOSUB 10000:GOSUB 9000
1100 GOTO 960:REM LEGGO IL PROSSIMO COMANDO
1200 O IF FI=1 THEN RETURN
7970 REM ROUTINE:CHIEDE IL NUMERO DELLA REGINA DA MUOVERE,SE E' POSSIBILE
7980 REM FARLO, CIOE' SE SI E' FERMATA
7990 REM LA REGINA MOSSA PRECEDENTEMENTE
8000 IF FI=1 THEN GOTO 8050
8001 POKE 214,23:PRINT
8010 PRINT "█COMANDO NON ESEGUIBILE█";
8020 FOR I=0 TO 1000:NEXT I
8030 POKE 214,23:PRINT
8040 PRINT "                          ";
8041 RETURN
8050 POKE 214,20:PRINT
8060 PRINT SPC(28);"REGINA DA"
8070 PRINT "▓";SPC(28);"MUOVERE? ";
8080 ZL=1:GOSUB 60000:PRINT
8090 REM CONTROLLO CHE IL VALORE LETTO SIA CORRETTO
8100 R=VAL(IN$)
8110 IF R<1 OR R>8 THEN GOTO 8050
8120 REM LA REGINA DA MUOVERE E' LA R
8130 POKE 214,20:PRINT
8140 PRINT SPC(28);"         "
8150 PRINT "▓";SPC(28);"          "
8160 REM HO CANCELLATO LE VECCHIE SCRITTE
8170 POKE 214,20:PRINT
8180 PRINT SPC(28);"█REGINA";R;"█"
8181 FI=0
8190 RETURN
8990 REM ROUTINE:POSIZIONA LO SPRITE R
9000 POKE VI+2*(R-1),SP(R,1)
9010 POKE VI+2*R-1,SP(R,2)
9020 RETURN
9990 REM ROUTINE:SPOSTO LA REGINA INDICATA DA R
10000 A=PEEK(197)
10010 IF A<>41 THEN GOTO 10030
10015 IF SP(R,2)>52 THEN SP(R,2)=SP(R,2)-24:RETURN:REM ALTO
10020 RETURN:REM NESSUNO SPOSTAMENTO
10030 IF A<>44 THEN GOTO 10040
10035 IF SP(R,2)<220 THEN SP(R,2)=SP(R,2)+24:RETURN:REM BASSO
10036 RETURN:REM NESSUNO SPOSTAMENTO
10040 IF A<>50 THEN GOTO 10050
10045 IF SP(R,1)<196 THEN SP(R,1)=SP(R,1)+24:RETURN:REM DESTRA
10046 RETURN:REM NESSUNO SPOSTAMENTO
10050 IF A<>42 THEN RETURN:REM NESSUNO SPOSTAMENTO
10055 IF SP(R,1)>28 THEN SP(R,1)=SP(R,1)-24:RETURN:REM SINISTRA
10060 RETURN:REM NESSUNO SPOSTAMENTO
10980 REM ROUTINE:METTE FUORI QUADRO LA REGINA R-ESIMA, SE E' LA ZERO NON
10990 REM SUCCEDE NIENTE
11000 POKE 214,20:PRINT:REM CANCELLO SCRITTA 'REGINA N'
11001 PRINT SPC(28);"          "
11009 SP(R,1)=220
11010 SP(R,2)=52+(R-1)*24
11020 POKE VI+2*(R-1),SP(R,1)
11030 POKE VI+2*R-1,SP(R,2)
11050 R=0:FI=1:CO=0
11060 REM CANCELLO EVENTUALE SCRITTA DI COLLISIONE
11070 POKE 214,23:PRINT
11080 PRINT "                          ";
11090 RETURN
11980 REM ROUTINE:CONTROLLA CHE NON CI SIANO COLLISIONI,CI SI ENTRA SOLO
11990 REM SE SI E' DEFINITA LA REGINA DA MUOVERE
12000 IF FI=0 THEN GOTO 12009
12001 REM SCRIVO 'COMANDO NON ESEGUIBILE'
12002 POKE 214,23:PRINT
12003 PRINT "█COMANDO NON ESEGUIBILE█";
12004 FOR I=0 TO 1000:NEXT I:REM RITARDO
12005 POKE 214,23:PRINT:REM CANCELLO LA SCRITTA
```

posizionare una regina, il programma chiama la routine 12000 (linea 990) che, posti in due variabili (X e Y) i valori delle coordinate della regina in gioco (linee 12090÷12100), esegue le istruzioni necessarie a determinare se questa si trova sotto scacco (linee 12110, 12120). Se si verifica uno scacco è presentata la scritta «collisione» (linea 12170) e si dovrà continuare a muovere la regina in gioco fino ad individuare una casella non minacciata. Quando si riesce a posizionare la regina il programma pone a zero la variabile R e ad uno FI, per permettere una successiva selezione (linea 12130). Dopo aver posizionato la regina è chiamata la routine 13000, che controlla se è stato completato il gioco. In tal caso è visualizzata una scritta lampeggiante (linee 13050÷13120) prima di disattivare gli sprites che rappresentano le regine (linea 13130) e di porre fine all'esecuzione del programma.
La routine che gestisce i comandi che permettono di muovere una regina sulla scacchiera è la 10000. Il programma la può chiamare solo se il valore della variabile R è diverso da zero, cioè solo se è stata selezionata una regina da mettere in gioco (linea 1010). La routine inizia caricando nella variabile A il codice del tasto che è stato premuto, e determina in base al valore di A se e in che direzione è stato richiesto uno spostamento. Sono quindi calcolate le coordinate della nuova posizione che dovrà assumere la regina

```
12006 PRINT "                              ";
12007 RETURN:REM TORNO ALLA FASE DI LETTURA COMANDI
12009 POKE 214,23:PRINT
12010 PRINT SPC(10);"            ";
12020 REM HO CANCELLATO L'EVENTUALE SCRITTA DI COLLISIONE
12021 CO=0:REM NON CI SONO COLLISIONI
12050 X=SP(R,1)
12060 Y=SP(R,2)
12070 J=1
12080 IF J=R THEN GOTO 12120
12081 IF SP(J,1)>196 THEN GOTO 12120
12090 X1=SP(J,1)
12100 Y1=SP(J,2)
12110 IF X=X1 OR Y=Y1 OR ABS(X-X1)=ABS(Y-Y1) THEN GOTO 12160:REM COLLISIONE
12120 J=J+1:IF J<9 THEN GOTO 12080
12130 R=0:FI=1
12131 REM CANCELLO LA SCRITTA 'REGINA N'
12132 POKE 214,20:PRINT
12133 PRINT SPC(28);"          "
12140 RETURN:REM NON CI SONO COLLISIONI
12150 REM C'E' UNA COLLISIONE
12160 POKE 214,23:PRINT
12170 PRINT SPC(10);"▒▒COLLISIONE▒▒";
12180 CO=1
12190 RETURN
12980 REM ROUTINE:SE LE OTTO REGINE SONO SULLA SCACCHIERA, DATE LE CONDIZIONI
12990 REM DI INGRESSO A QUESTA ROUTINE, IL SOLITARIO E' TERMINATO
13000 I=1
13010 IF SP(I,1)=220 THEN RETURN
13020 I=I+1:IF I<9 THEN GOTO 13010
13030 REM IL SOLITARIO E' TERMINATO
13040 REM FACCIO LAMPEGGIARE
13050 FOR I=1 TO 100
13060 POKE 214,23:PRINT
13070 PRINT "BRAVO, HAI TERMINATO IL SOLITARIO.";
13080 FOR J=1 TO 20:NEXT J
13090 POKE 214,23:PRINT
13100 PRINT "▒BRAVO, HAI TERMINATO IL SOLITARIO.▒";
13110 FOR J=1 TO 20:NEXT J
13120 NEXT I
13130 POKE VI+21,0
13131 POKE 53280,14:POKE 53281,6
13132 PRINT "▒▒▒"
13140 END
40990 REM REGINA 1
50000 DATA 0,0,0,0,128,0,1,192,0,3
50010 DATA 96,0,30,60,0,16,4,0,16,4
50020 DATA 0,49,134,0,96,131,0,192,129,128
50030 DATA 96,131,0,48,134,0,17,196,0,16
50040 DATA 4,0,30,60,0,3,96,0,1,192
50050 DATA 0,0,128,0,0,0,0,0,0,0
50060 DATA 0,0,0,0
50061 REM REGINA 2
50070 DATA 0,0,0,0,128,0,1,192,0,3
50080 DATA 96,0,30,60,0,16,4,0,16,4
50090 DATA 0,48,198,0,97,35,0,192,33,128
50100 DATA 96,67,0,48,134,0,17,228,0,16
50110 DATA 4,0,30,60,0,3,96,0,1,192
50120 DATA 0,0,128,0,0,0,0,0,0,0
50130 DATA 0,0,0,0
50131 REM REGINA 3
50140 DATA 0,0,0,0,128,0,1,192,0,3
50150 DATA 96,0,30,60,0,16,4,0,17,196
50160 DATA 0,50,38,0,96,35,0,192,65,128
50170 DATA 96,35,0,50,38,0,17,196,0,16
50180 DATA 4,0,30,60,0,3,96,0,1,192
50190 DATA 0,0,128,0,0,0,0,0,0,0
50200 DATA 0,0,0,0
50201 REM REGINA 4
50210 DATA 0,0,0,0,128,0,1,192,0,3
50220 DATA 96,0,30,60,0,16,4,0,16,132
50230 DATA 0,49,6,0,97,3,0,194,1,128
50240 DATA 98,3,0,52,70,0,23,228,0,16
50250 DATA 68,0,30,60,0,3,96,0,1,192
50260 DATA 0,0,128,0,0,0,0,0,0,0
50270 DATA 0,0,0,0
50271 REM REGINA 5
50280 DATA 0,0,0,0,128,0,1,192,0,3
50290 DATA 96,0,30,60,0,16,4,0,19,228
50300 DATA 0,50,6,0,98,3,0,193,193,128
50310 DATA 96,35,0,50,38,0,17,196,0,16
50320 DATA 4,0,30,60,0,3,96,0,1,192
50330 DATA 0,0,128,0,0,0,0,0,0,0
50340 DATA 0,0,0,0
50341 REM REGINA 6
50350 DATA 0,0,0,0,128,0,1,192,0,3
50360 DATA 96,0,30,60,0,16,4,0,17,196
50370 DATA 0,50,38,0,98,3,0,193,193,128
50380 DATA 98,35,0,50,38,0,17,196,0,16
50390 DATA 4,0,30,60,0,3,96,0,1,192
```

aggiungendo o sottraendo 24 alle coordinate della posizione attuale (24 è il numero di bits che separa i centri di due caselle adiacenti; infatti ogni casella è costituita da 3×3 caratteri e ogni carattere da 8×8 bits). Si sottrae 24 alla coordinata Y per determinare uno spostamento di una casella verso l'alto (linea 10015), e lo si somma alla medesima coordinata per uno spostamento verso il basso (linea 10035). Per effettuare uno spostamento verso destra si somma 24 alla coordinata X (linea 10045) e lo si sottrae per effettuarlo verso sinistra (linea 10055). Il programma poi chiama nuovamente la routine 9000 per visualizzare lo sprite nella nuova posizione.

```
50400 DATA 0,0,128,0,0,0,0,0,0,0
50410 DATA 0,0,0,0
50411 REM REGINA 7
50420 DATA 0,0,0,0,128,0,1,192,0,3
50430 DATA 96,0,30,60,0,16,4,0,17,244
50440 DATA 0,48,22,0,96,35,0,192,65,128
50450 DATA 96,67,0,48,134,0,16,132,0,16
50460 DATA 4,0,30,60,0,3,96,0,1,192
50470 DATA 0,0,128,0,0,0,0,0,0,0
50480 DATA 0,0,0,0
50481 REM REGINA 8
50490 DATA 0,0,0,0,128,0,1,192,0,3
50500 DATA 96,0,30,60,0,16,4,0,17,196
50510 DATA 0,50,38,0,98,35,0,193,193,128
50520 DATA 98,35,0,50,38,0,17,196,0,16
50530 DATA 4,0,30,60,0,3,96,0,1,192
50540 DATA 0,0,128,0,0,0,0,0,0,0
50550 DATA 0,0,0,0
59920 REM ROUTINE:GESTISCE L'INPUT DI UNA STRINGA ALFANUMERICA
59930 REM LE VARIABILI UTILIZZATE SONO:Z6,Z7,Z8,Z9,Z8$,IN$
59940 REM IN INGRESSO VUOLE IL VALORE ZL CHE E' LA LUNGHEZZA MASSIMA DELLA
59950 REM STRINGA DA LEGGERE
59960 REM IN USCITA DA' LA STRINGA LETTA IN IN$ E LA SUA LUNGHEZZA IN Z9
59980 REM CANCELLO LA ZONA DI SCHERMO IN CUI VERRA' DIGITATA LA STRINGA
60000 FOR Z8=1 TO ZL: PRINT " ";:NEXT Z8
60010 FOR Z8=1 TO ZL: PRINT "▌";:NEXT Z8
60020 IN$="":Z7=TI
60040 REM LEGGO UN CARATTERE
60060 GET Z8$:IF Z8$<>"" THEN 60160
60080 REM ACCENSIONE E SPEGNIMENTO DEL CURSORE
60100 IF Z7<TI AND NOT(Z6) THEN PRINT "▒▌";:Z6=NOT(Z6):Z7=TI+15
60110 IF Z7<TI AND Z6 THEN PRINT " ▌";:Z6=NOT(Z6):Z7=TI+15
60120 GOTO 60060
60140 REM E' STATO DIGITATO UN CARATTERE
60160 Z8=ASC(Z8$):Z9=LEN(IN$)
60180 REM SE NON E' UN CARATTERE ALFANUMERICO, DEVE ESSERE UN RETURN O DELETE
60200 IF NOT((Z8>47 AND Z8<58) OR (Z8>64 AND Z8<91)) THEN GOTO 60320
60220 REM CONTROLLO CHE NON SIA STATA SUPERATA LA LUNGHEZZA MASSIMA
60240 IF Z9=ZL THEN GOTO 60060
60260 REM LO AGGIUNGO ALLA STRINGA IN$
60280 IN$=IN$+Z8$:PRINT Z8$;:GOTO 60060
60300 REM SE E' UN RETURN, HO TERMINATO LA LETTURA
60320 IF Z8=13 THEN PRINT " ▌";:RETURN
60340 REM SE E' DELETE, CANCELLO IN IN$ E SUL VIDEO L' ULTIMO CARATTERE DIGITATO
60360 IF Z8=20 AND Z9>0 THEN IN$=LEFT$(IN$,Z9-1):PRINT " ▌▌";:GOTO 60060
60370 GOTO 60060
60960 REM ROUTINE:INIZIALIZZAZIONE COSTANTI
60980 REM CIRCUITO VIDEO
61000 VI=53248
61020 REM CIRCUITO SUONO
61040 SI=54272
61060 REM MEMORIA VIDEO
61080 MV=1024
61100 REM MEMORIA COLORE
61120 MC=55296
61140 REM COSTANTI DI USO COMUNE
61160 ZL=9
61180 REM INIZIALIZZAZIONE CHIP SUONO
61200 FOR I=0 TO 24
61210 POKE SI+I,0
61220 NEXT I
61230 RETURN
61980 REM ROUTINE:STAMPA IL TITOLO DEL PROGRAMMA MEMORIZZATO IN PG$
62000 GOSUB 61000
62010 POKE VI+32,15
62020 POKE VI+33,15
62030 PRINT "▒▒▒▒▒▒";
62040 PRINT TAB(6);"┌──────────────────────────┐"
62050 FOR I=1 TO 5
62060 PRINT TAB(6);"|                          |"
62070 NEXT I
62080 PRINT TAB(6);"└──────────────────────────┘"
62090 PRINT TAB(6);"▒▒▒▒▒▒▒DIGITA ▒RETURN▒ PER PROSEGUIRE"
62100 PRINT "▒▒▒▒▒▒"
62110 FOR I=1 TO 5
62120 PRINT TAB(7);
62130 FOR J=1 TO 26
52140 PRINT "▒▒ ";
52150 NEXT J
62160 PRINT
62170 NEXT I
62190 REM ORA SCRIVO IL TITOLO
62210 PRINT "▒▒▒▒▒▒▒▒▒";TAB((40-LEN(PG$))/2);"▒▒";PG$
62220 GET Z9$
62230 IF Z9$<>CHR$(13) THEN GOTO 62220
62240 PRINT "▒▒";
62250 RETURN
```

INDICE

Il personal computer Commodore 64 pagg. 7-37

La tastiera 8-12
Tasti di scrittura 9
Tasti grafici 9
Tasti di controllo 10-12
Tasti funzionali 12

Il video 13-22
Il cursore 13-14
Il colore 14-16
Le mappe di memoria del video 16-18
La grafica 18-22

Il suono 23-24

Il Basic del CBM-64 25-37
L'interprete 25-26
Dati, risultati, costanti e variabili 26-27
Gli operatori 27-28
I comandi 28-29
Le istruzioni 29-35
Le funzioni 35-37

I programmi (* di Franco Arcieri - ** di Pierfrancesco Armati) 39-191

Poker d'assi * 39-45
Un medico poco serio ** 46-56
Computo, computas... ** 57-68
Strike and ball * 69-76
La torre di Hanoi * 77-83
Grafica monodimensionale * 84-89
Corsa d'auto * 90-95
Alta risoluzione * 96-101
Disegna il tuo sprite * 102-108
Grafica tridimensionale * 109-115
Slot machine * 116-121
Proviamo con l'Assembler * 122-131
Labirinto * 132-139
Il gioco della vita * 140-146
Memory dump * 147-152
Archivio * 153-160
Aritmetica a precisione infinita * 161-168
L'archivio veloce * 169-176
Editor * 177-185
Le otto regine * 186-191

Hanno collaborato all'illustrazione del volume: il Centro THC (pag. 1), la Commodore italiana (pagg. 6 e 38), Dominique Fradin e Patrizia Savarese (copertina), Vincenzo Pirozzi (pagg. 13 ÷ 16 e 39 ÷ 186).

Finito di stampare nel novembre 1984
presso OFSA - Casarile (Milano)

**20 programmi
assolutamente nuovi
ed inediti
per disegnare, calcolare,
scrivere, giocare,
catalogare, ricordare
e studiare col computer...
ma soprattutto
per imparare
come funziona
e come si programma
il tuo Commodore 64.**

L. 8000